CIENCIAS DE LA COMUNICACIÓN Y DISEÑO

Introducción a los Sistemas Multi-Agente

Parte I

Wulfrano Arturo
Luna Ramírez

Casa abierta al tiempo
UNIVERSIDAD AUTÓNOMA METROPOLITANA

TECNOLOGÍAS DE LA INFORMACIÓN

COLECCIÓN LIBRO DE TEXTO

Introducción a los sistemas multiagente

Parte 1

Wulfrano Arturo Luna Ramírez

Esta obra es una de las ganadoras de la “Convocatoria para libros de texto como apoyo en la impartición de los programas de estudios”, 2024. Fue evaluada para su publicación por el Consejo Editorial de la UAM Unidad Cuajimalpa, con base en los dictámenes solicitados a pares académicos mediante un esquema que preserva el anonimato mutuo. Estos dictámenes resultaron favorables.

Diseño de portada: Aldo Juárez Herrera
Corrección de estilo: Mariana Aguilar Mejía
Ilustraciones: Sara Margarita Bustamante Loya

Universidad Autónoma Metropolitana
Unidad Cuajimalpa
Av. Vasco de Quiroga 4871, col. Santa Fe
Alcaldía Cuajimalpa de Morelos
C.P. 05348, Ciudad de México

www.cua.uam.mx

ISBN: 978-607-28-3312-8
ISBN: 978-607-28-3020-2 (Colección)

Agradecimientos

La ruta que siguió este trabajo fue por demás sinuosa, fue un camino acompañado por varias personas cuyo apoyo moral permitió que este esfuerzo siguiera su curso.

Por ello, agradezco a mis profesores del posgrado en Inteligencia Artificial de la Universidad Veracruzana, especialmente a los Doctores Manuel Martínez Morales, José Negrete Martínez, y la Maestra Angélica Gacía Vega, sus fundadores. Ellos supieron tejer narrativas para explicarme a mi y a muchos otros los temas fundamentales de esta disciplina, aún en ciernes. También pudieron ilustrar situaciones donde la aplicación de la Inteligencia Artificial, de sus métodos y tecnologías abona a la consecución de una respuesta a los misterios y dificultades que nos exige la vida diaria, así como de nuestras necesidades e inquietudes dentro y fuera de la disciplina. Por supuesto agradezco a mis maestros, los Doctores Alejandro Guerra, Nicandro Cruz, Héctor Acosta, Antonio Marín, Fernando Montes y Guillermo Hoyos.

También agradezco a mis colegas del Departamento de Tecnologías de la Información de la División de Ciencias de la Comunicación y Diseño/Universidad Autónoma Metropolitana-Cuajimalpa, particularmente a los Doctores Christian Lemaître y Héctor Jiménez. Así mismo, agradezco a mi mentora del doctorado en la Universidad de Essex, la Doctora Maria Fasli.

Además, va mi gratitud al personal administrativo y de apoyo que posibilita la operación de universidades, departamentos y de los mismos posgrados en las instituciones mencionadas.

Dedicatoria

A Esther, por dar la vuelta al mundo
en aras del encuentro

A Perfecta y Wulfrano, por cultivar las semilla
y dejarme cosechar los frutos

A Toño, María y Francisco,
en las sonrisas nos reconocemos

A mis familiares y amigos, por compartir la vida

Prefacio

La Inteligencia Artificial ha devenido en la última década en una materia de revisión obligada para estudiantes de Informática, Tecnologías de la Información y Computación en general, académicos de otras áreas y público no especializado. Lo anterior se origina por las múltiples aplicaciones que se han dado a conocer tanto en la industria como en otros sectores. No es fortuita su popularidad actual; sin embargo, a veces parece que es una disciplina más bien opaca y destinada a sólo ciertos sectores selectos de personas o instituciones. Este libro pone de manifiesto lo equívoco de esta concepción.

De esta forma, el material aquí presentado ofrece una introducción al área, revisa sus fundamentos y proporciona material para adentrarse en su estudio desde la óptica de los Agentes Artificiales y los Sistemas Multi-Agente, además de que sirve de vehículo para la exploración de textos más avanzados.

Se dispone en él un enfoque teórico-práctico dentro de un contexto cercano a los lectores a quienes se dirije, que son en primer lugar alumnos de la Licenciatura en Tecnologías y Sistemas de Información (LTSI), pues apoya de forma predominante a las Unidad de Enseñanza Aprendizaje (UEA) siguentes:

- Inteligencia Artificial I
- Seminario de Sistemas Inteligentes I
- Seminario de Sistemas Inteligentes II

Adicionalmente, pueden beneficiarse de él los siguientes cursos de las UEA: Seminario de Tecnologías de la Información I y II; Seminarios de Sistemas de Información I y II; y muy plausiblemente también los Laboratorios Temáticos I a IV. Todas impatidas por el Departamento de Tecnologías de la Información, perteneciente a la División de Ciencias de la Comunicación y Diseño de la Universidad Autónoma Metropolitana, Unidad Cuajimalpa.

Aunado al cumplimiento del cometido antedicho, este texto será de utilidad al público interesado en intoducirse al mundo de los sistemas inteligentes, particularmente de los agentes artificiales y la simulaciones que se pueden conseguir con ellos.

El material didáctico que se presenta en este trabajo ofrece los fundamentos para programar sistemas inteligentes usando el paradigma de agentes artificiales. Para ello, se presentan problemas con su solución a través de un diseño de agentes y su respectiva implementación.

La información que se expone en el capitulado asume un conocimiento básico en computación, de editores de textos y manejo de archivos, y requiere del conocimiento elemental de la programación de computadoras. Es claro que quienes lo

tengan podrán beneficiarse mayormente del contenido aquí propuesto, pero no es limitativo: la parte teórica se puede seguir aún prescindiendo de este bagaje.

Los temas que se presentan en su parte práctica se desarrollan mediante una exposición de los fundamentos y la ejemplificación sencilla de los principales postulados que los subyacen.

En un inicio, se presentan las ideas clave de la Inteligencia Artificial, sus métodos y principales logros, así como sus limitaciones. A partir del segundo capítulo, se introduce el enfoque de Agentes Atificiales en esta disciplina. Posteriormente, se presenta la concepción de Sistemas Multi-Agente; para terminar con la temática relacionada con la Simulación Basada en Agentes, que permite redondear las ideas expuestas en los temas que la anteceden desde un punto de vista integrador.

El libro culmina con una reflexión de la dimensión humana en la Inteligencia Artificial y los Sistemas Multi-Agente.

Las ilustraciones fueron realizadas por Sara Margarita Bustamante Loya, a quien le agradezco el esfuerzo y la dedicación para este cometido.

Índice

Capítulo 1

Inteligencia... aunque sea artificial

Objetivos

- Conceptualizar el término Inteligencia Artificial
- Hacer un breve recuento histórico del área
- Introducir el enfoque de Agentes Inteligentes

El placer es, para el ser humano, el ejercicio de la razón.
Aristóteles

1. Génesis de la Inteligencia Artificial

1.1. La cuestión de la Inteligencia

Desde tiempos inmemoriales, el ser humano se ha preguntado, admirado del mundo en el que está circunscrito, sobre el sentido de su ser, sobre su naturaleza y la naturaleza de las cosas. Como una de las primeras preguntas se cuestiona el ¿por qué?, ¿por qué se vive y por qué se muere? A continuación se plantea el ¿para qué?, ¿para qué la vida? Consecuentemente ¿para qué *nuestra* vida? Es la inquietud por saber la que despierta estas y muchas otras interrogantes que en todas las culturas se manifiestan y se responden de diversas maneras. Por ejemplo, en la tradición intelectual occidental, los primeros filósofos griegos se preguntan por el *arché*, es decir, sobre aquello que origina o gobierna todas las cosas [1]. A raíz de estos cuestionamientos, surge aquel que se refiere a nuestra propia "forma de hallar esas respuestas": ¿cómo es que somos capaces de preguntarnos sobre las cosas y plantear respuestas? La humanidad se vuelve así su propio objeto de estudio al inquirir sobre su capacidad de pensamiento: ¿cuál es el *arché* del pensar humano? De esta forma, el ser humano se identifica como un ser pensante, racional, incluso se denomina *Homo sapiens* (hombre sabio), en un intento por subrayar sus capacidades mentales [2].

La actividad intelectual del ser humano se ve reflejada en su accionar diario bajo diversas formas, partiendo de su lucha por la supervivencia hasta la creación de la cultura: la fabricación de artefactos y máquinas, el surgimiento del arte, de organizaciones sociales, de la escritura, etc. De ahí la importancia de la reflexión sobre el pensamiento, pues el humano llega a hacer del pensar el mismo objeto del pensar: se convierte en su propio objeto de estudio. He aquí las bases de muchas disciplinas, dentro de las que ubicamos al estudio denominado Inteligencia Artificial.

1.2. El concepto de inteligencia

Una fuente de desacuerdo entre varias áreas es el concepto de *inteligencia*. No hay un consenso al respecto, entre otras razones, porque se trata de un término descriptivo, acerca de ciertas propiedades de individuos o grupos, y al ser descriptivo es arbitrario, pues cada quien lo asocia con aquello que considera relevante. De tal manera, ninguna definición particular puede satisfacer a todos [3]. Sin embargo, de forma general, todos "sabemos" lo que es, inclusive podemos relacionarle con algunas actividades [3], por ejemplo, reconocer una persona, llamar por teléfono, conducir un auto, escuchar (y entender, identificarlo, reírse de) un mal chiste, caminar hacia un destino específico, comprar un periódico, una computadora, jugar algún juego de mesa, planear la cena de fin de año, cocinar un platillo, entre muchas otras. Además, hay dos conceptos relacionados con la inteligencia de forma más o menos generalizada: la innovación y la adaptabilidad, las cuales sirven de base para acercarse, si no a una definición común, sí al entendimiento de la inteligencia. Con el fin de plantear

un concepto y ganar un poco de especificidad, en este trabajo se hará referencia a la inteligencia a partir del siguiente enunciado [1]:

Concepto 1 *Inteligencia (del latín* intellego*, liga entre, entender): es la capacidad de resolver problemas de orden intelectual, moral o vital en un tiempo determinado.*

De este concepto, intuitivamente se desprenden varias preguntas ¿tienen los animales la capacidad de pensar?, ¿son inteligentes?, ¿cómo medimos la inteligencia?, ¿es una capacidad heredada o es adquirida?, ¿qué papel juegan las emociones en la inteligencia?, ¿son antítesis las unas de la otra?, y nuestra preguntas favoritas: ¿cómo contiende el cerebro o la mente con los problemas que el mundo le presenta?, ¿la máquinas pueden pensar?, ¿podemos construirlas, bajo qué principios?

1.3. ¿Qué se entiende por Inteligencia Artificial?

La idea de construir dispositivos que hagan tareas de forma similar al ser humano es vieja. Muestra de ello son las historias sobre seres extraordinarios creados por el hombre mediante magia negra, intervención divina y artificios científicos de los cuales se tienen varios registros, principalmente literarios. En muchas culturas se presentan leyendas que hablan de estos seres y artefactos, desde androides hasta robots: los titanes griegos, Talos, Hefesto, los gólems, los autómatas de relojería europeos, así como dispositivos chinos y de medio oriente antiguo, etc. En América, según lo mencionan atinadamente tanto el Dr. José Negrete Martínez [4] como el Dr. Manuel Martínez [5], tenemos en el Popol Vuh de los mayas quiché [6] un ejemplo bastante interesante de una explicación creacionista del ser humano, que podría ajustarse a lo que se considera un experimento *gedanken*[1], veamos el siguiente fragmento:

> ... de lodo hicieron la carne [del hombre]. Pero vieron que no estaba bien, porque se deshacía, estaba blando, no tenía movimiento, no tenía fuerza, se caía, estaba aguado, no movía la cabeza, la cara se le iba para un lado, tenía velada la vista, no podía ver hacia atrás. Al principio hablaba, pero no tenía entendimiento. ... ¿Cómo haremos para perfeccionar, para que salgan bien nuestros adoradores, nuestros invocadores?

Conforme a la narración, el Popol Vuh —nos cuenta el Dr. Negrete [4]—, explica entre otras cosas cómo se da el desarrollo y buen fin de la síntesis de "los adoradores con entendimiento" (¿máquinas?, ¿robots?). Tal cometido se efectúa pasando por varias fases incrementales donde se exhiben elementos de diseño de tipo diverso: 1) Se intenta hacer una máquina inteligente con materiales adecuados, pero se descubre que hay un avance sí, pero no se logra sólo con los materiales y la energía de la naturaleza. 2) Mediante experimentación incremental, se puede lograr una máquina inteligente: máquinas autoenergizadas no autónomas; máquinas autónomas sin programación; máquinas programables incomunicadas;

[1] Un experimento Gedanken (del alemán *gedankenexperiment*) es un experimento mental propuesto para probar alguna teoría o hipótesis que no puede hacerse de manera física debido a limitaciones prácticas. Dos ejemplos son el gato de Schrödinger y el demonio de Maxwell. Cfr.: http://en.wikipedia.org/wiki/Thought_experiment y http://plato.stanford.edu/entries/thought-experiment/.

máquinas dialogantes sin entendimiento; máquinas con entendimiento limitado; máquinas con entendimiento amplio pero ambiguo, y finalmente máquinas no ambiguas pero con entendimiento ponderadamente reducido.

De acuerdo con la perspectiva de Haugeland [7], existen dos tipos de artefactos o mecanismos inteligentes que la humanidad ha recogido en sus registros y que están presentes en el imaginario popular (por lo menos en la tradición occidental, que es a la que él pertenece y de la que nos habla):

- **Zoomorfos y antropomorfos**: construidos por el hombre, exhiben superioridades y causan aversión por su aspecto deformado y su actuar malévolo. Recordemos a la criatura de Frankenstein, Hefesto, gólem, etc. Son producto de la ciencia, la magia negra y el misterio.
- **Robots mecánicos**: son resultado del desarrollo de la "alta tecnología", por lo regular ruidosos, con mecanismos complicados, generalmente son producto de las proyecciones que inspiraron los relojes de cuerda, las máquinas de vapor, los telares, las centrales telefónicas, entre otros.

Como se puede apreciar, la idea de sintetizar la actividad intelectual no es nueva: varios pensadores han dilucidado de manera más profunda sobre el particular; tal es el caso de Llull, Descartes y Leibnitz, quienes intentaban la mecanización total del razonamiento humano [8], y particularmente los trabajos de Hobbes (quien postula que razonar no es otra cosa que ejecutar un proceso mecánico de cómputo) y Hume, quien intentara construir una mecánica mental al estilo de Newton [2, 4]. Sin embargo, no fue sino con la llegada de las computadoras que la idea renace con más ímpetu. Veamos el supuesto siguiente:

$$\text{Pensar (la actividad intelectual)} = \text{Una } \textit{manipulación de símbolos} \text{ mentales (ideas)}$$

Es claro que, si se acepta esto, y si se considera que las computadoras actuales hacen una manipulación de símbolos, se puede obtener algo parecido a la manipulación que lleva a cabo la mente; luego, puede contarse con máquinas con mente propia, máquinas pensantes: tal es el origen de la Inteligencia Artificial (IA) [7].

Antes de intentar dar una noción más precisa sobre la IA, conviene mencionar una campo de conocimiento que ha intentado unificar los estudios de varias disciplinas en torno a los principios abstractos de la organización mental, como son la Psicología, la filosofía de la mente, y la propia IA: la ciencia cognitiva, que pretende ser un estudio coherente del conocimiento, la inteligencia y la mente, esto es, de la manipulación de símbolos, atendiendo al supuesto

```
pensar = computar
```

Dentro de este campo más general, podemos pensar que la IA realiza este estudio usando el punto de vista informático o de las ciencias de la computación, valiéndose de sus herramientas y técnicas. Sin embargo, hay que señalar que no se circunscribe únicamente a la computación y la informática: emplea dichas herramientas, pero no se limita a ellas. Consecuentemente, algunos autores consideran entonces que la IA es el estudio de la inteligencia natural por medios artificiales[2].

[2]Definición manifestada por el Dr. José Negrete, entre otros, oída por el autor de manera presencial.

Es necesario resaltar que tal estudio se propone además de entender, reproducir la inteligencia (o si se quiere, instanciarla mediante entidades artificiales).

Por otro lado, existen varios enfoques en IA [2]: aquellos que se plantean como objetivo central la conducta de la máquina, les interesa la forma en que ésta actúa mientras que otros buscan máquinas pensantes. Por otro lado, hay quienes se preocupan por los procesos mentales y el razonamiento; de esta forma, intenta obtener sistemas que piensen-actúen como el humano. En tanto que otro grupo de investigadores se centra en el concepto de racionalidad, sea ésta humana o de otro tipo.

En la figura 1.1 se presentan estos enfoques.

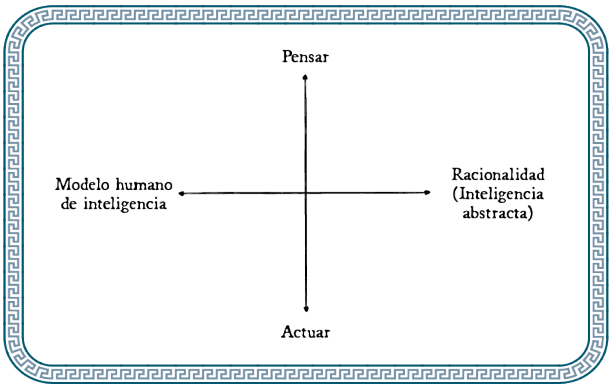


Figura 1.1: Enfoques de la IA. Sistemas orientados a la conducta o al pensamiento, tomando como base al ser humano o a un modelo de inteligencia abstracta. Adaptado de [2]

La prueba de Turing y el cuarto chino

Partiendo de la panorámica expuesta anteriormente, a continuación se menciona una de las propuestas más importantes sobre el concepto y la posibilidad de la IA. Alan Turing (1912-1954) postuló una prueba para definir operativamente la inteligencia: una conducta es inteligente si logra una eficiencia comparable con la del humano en todas las actividades cognitivas, suficiente para engañar a un evaluador. La prueba de Turing se basa en el juego de la imitación, el cual plantea, grosso modo, que dos personas sostengan una conversación por teletipo y una de ellas sea sustituida por una computadora –con las condiciones materiales para que la otra no se dé cuenta de la sustitución [3].

Si la máquina es capaz de dialogar como lo haría una persona en las mismas circunstancias, sin que el humano lo identifique, entonces esa máquina puede considerarse poseedora de una conducta inteligente. Sostener una conversación implica que la máquina debe exhibir buenos conocimientos del idioma y una comprensión aceptable del tema (sea de política, gastronomía, poesía, deportes, etc.), y esto no es trivial para una máquina.

[3] Para una visión actualizada, piénsese que las condiciones serían parecidas a las existentes cuando una persona dialoga con otra mediante un programa de mensajería instantánea —comúnmente conocido como *chat*—, con o sin intercambio de voz (en el segundo caso, usando un sintetizador de voz de alta fidelidad), o con capacidades robustas de generación de video o sin él, o bien, únicamente mediante texto.

Como es de esperar, existen varias objeciones a esta prueba, las cuales no serán abordadas en este texto para mantener la brevedad, pero que el lector interesado puede consultar en las referencias.

Dentro de estas objeciones destaca "el cuarto chino" de John Searle [2, 7, 8]. Hay que resaltar que la idea de la prueba es poner de manifiesto los aspectos cognitivos de la cuestión (estructura y operaciones internas que permiten que un sistema diga lo correcto en tiempo y forma).

1.4. El dictum de Ada Lovelace

Para cerrar esta sección se menciona una de las objeciones más importantes y recurrentes acerca de la posibilidad de que las computadoras exhiban inteligencia: el díctum de Lady Ada Augusta Lovelace [7], expresado *grosso modo* como sigue:

> Una computadora sólo puede hacer lo que el programador le ha indicado, nada más.

Dentro de las consecuencias de esta afirmación se encuentra la imposibilidad de que las máquinas sean inteligentes, de que ostenten libertad (libre albedrío, control propio), creatividad y responsabilidad, entre muchas otras. Los intentos por vulnerar este díctum se plantean en dos planos: una vulneración en principio y una pragmática. Partiendo de lo anterior, se presentan las siguientes consideraciones [9, 7]:

- En primer lugar, hay que preguntarse ¿qué es un programa? En términos generales, es la especificación precisa de los procesos internos aplicables de una máquina (orientada a alguna tarea). En este sentido, también se considera la implementación de un algoritmo en una máquina. Ahora, hay que resaltar que algunos han postulado lo siguiente: "estar programado para" no es incompatible con ser libre, creativo o algo más, pues el programa es resultado de un diseño, y si éste incluye cuestiones afines (estar diseñado para la creatividad), entonces no hay contradicción al afirmar que se está "programado para la creatividad". Por último, se menciona que gracias al área de **Aprendizaje de Máquina** o **Automático**, aquello que se programa de forma directa es la información y los principios generales de operación —-como cuando asistimos a la escuela—. Lo que las computadoras hacen con ello no es programado directamente ni fácilmente predecible por el diseñador-programador (siguiendo el símil, la actuación posterior de los educandos no la determinan los profesores, sino ellos mismos). Esto se menciona con más énfasis en el siguiente punto.

- Las respuestas ofrecidas por la IA, con mayor o menor suerte, son todas **intentos pragmáticos**. Entre ellos tenemos:

  a) si los símbolos de la IA clásica pudieran asociarse entre sí, sin la intervención de un supervisor, estaríamos ante la cuestión de la existencia de otra inteligencia, además de la nuestra;

  b) la búsqueda exitosa de funciones de evaluación —desconocidas por el programador—- para desempeñarse bien en determinadas tareas, como el juego de damas, el Go, u otro más;

c) el enfoque del Cómputo Bioinspirado, como lo que reflejan los Algoritmos Genéticos, que hacen una búsqueda en un espacio de patrones, representa otra posibilidad; pues la manera de los seres vivos de entender el mundo es manejando patrones. De tal forma, una máquina es capaz de crear nuevas máquinas y, por ende, nuevas soluciones (no previstas por el programador).

- Finalmente, se habla de las **máquinas autopoiéticas** (inexistentes aún), las cuales incorporan el mundo de manera parcial, pero de forma tal que sus estados internos son indistinguibles de ese mundo. Un ejemplo de máquina autopoyética lo tendríamos en una red neuronal que ajustara sus pesos continuamente durante su entrenamiento y durante la solución de problemas; aunado a esto, dicha solución debería modificar el mundo, y este mundo modificado, a su vez, ser una entrada de la red. Las máquinas autopoiéticas se consideran la propuesta más prometedora para **vulnerar el díctum en principio**, no sólo de forma pragmática, sino teórica

A continuación, se hablará de la génesis de la IA como campo de estudio moderno.

2. El nacimiento de la Inteligencia Artificial

El nacimiento de la IA, como disciplina moderna de estudio, se ubica entre los años 1943 y 1956, periodo en el que se publican los primeros trabajos del área como tal (redes neurales, jugadores de ajedrez, demostradores de teoremas, entre otros), principalmente con el trabajo de Alan Turing acerca de la mente, identificado como el texto seminal de la IA: *Computing machinery and intelligence*.

Formalmente, la IA nace —-con ese nombre— en la mítica conferencia del Dartmouth College en el verano de 1956, donde se reunieron durante dos meses McCarthy (quien propuso el nombre), Minsky, Shannon, Rochester, Samuel, Solomonoff, Selfridge, Newell y Simon, entre otros investigadores. Esta conferencia produjo, además del nombre, una fuente de relaciones entre los investigadores, quienes realizaron importantes trabajos, muchos de los cuales aportaron ideas vigentes hasta la fecha.

A partir de ese momento, el campo de la IA floreció a causa del aumento de investigadores y presupuestos gubernamentales y privados, animados por los paulatinos logros que fueron obtenidos en distintas áreas. Sin embargo, muchos de los programas no funcionaban adecuadamente al cambiar de dominio de aplicación, o bien, cuando a los problemas a los cuales se aplicaban se les aumentaba la complejidad, aunque fuera en una o dos variables. En ese momento, el optimismo inicial fue bajando de tono y muchas de las aseveraciones temporales al estilo "en n años las máquinas podrán hacer x tareas" fueron mesurándose en su alcance y temporalidad. Se tuvo que reconocer que el mundo es mucho más complicado de lo que aparenta. De esta manera, se comprendió que **pese a que en principio, un programa sea capaz de encontrar una solución, esto no implica que se disponga de todos los mecanismos necesarios para encontrar la solución en la práctica** [2].

En fin, el campo de la IA ha producido éxitos impresionantes, optimismo sobredimensionado, fracasos y un proceso de mejora de las ideas más prometedoras

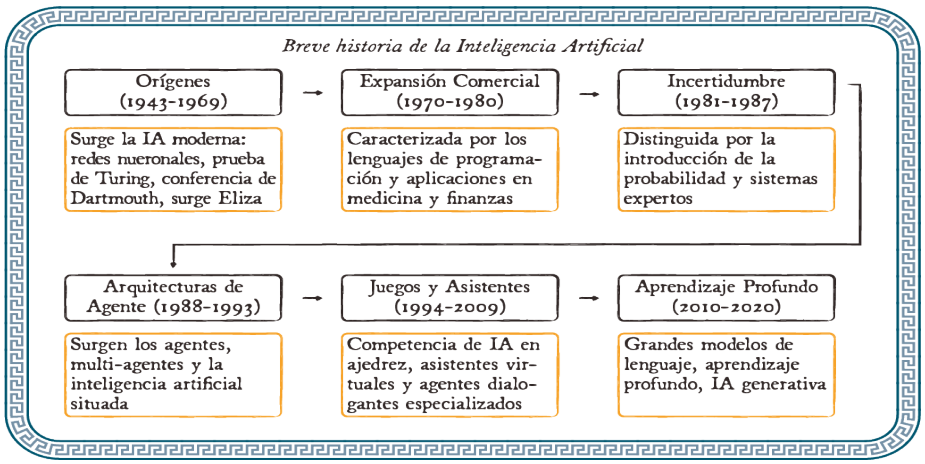


Figura 1.2: Breve historia de la IA y áreas o campos relevantes

producidas por los investigadores, cristalizadas (y no pocas veces, motivadas) por la industria y la relación con otras áreas. En la figura 1.2 se presenta un esquema resumen con algunos de los momentos más notorios en IA. Como nota a destacar, algunos de los hitos más conocidos en la década de 2020 pueden agruparse alrededor de dos rubros:

- Juegos
  - Deepstack gana a humanos en juegos de información imperfecta como el póker.
  - AlphaGo Zero - jugando sólo contra sí mismo derrotó a otras versiones (Lee y Master) y aprendió ajedrez en cuatro horas.
- Procesamiento de Lenguaje Natural/Visión Artificial
  - Algoritmos capaces de modificar videos y generar retratos de personas ficticias de forma realista; Google Duplex, asistente de toma de citas, es reconocido como una imitación casi sin errores del discurso humano.
  - ChatGPT de OpenAI y sus contrapartes como Bard y Claude han despertado mucho interés por su generación de texto bien formado (comparable con la de seres humanos) y sus extensiones multimodales.
  - Explosión de herramientas generativas para crear de texto, y, por extensión, de código fuente de programación.
  - Generación de imágenes y videos a partir de descripciones textuales.

En las secciones siguientes, se presenta un breve comentario sobre la IA Clásica y un esbozo de lo que se ha denominado la Nueva IA, así como un comentario sobre lo que se entiende hoy en día por IA o IA Hegemónica y uno de sus vástagos más publicitados, la IA Generativa.

2.1. La IA Clásica

Se ha denominado IA Clásica al enfoque que se basa en el supuesto de la importancia substancial que tiene el razonamiento deliberativo en la inteligencia, el cual conlleva la habilidad de reflexionar conscientemente acerca de un problema. Ésta es la primera aproximación de la IA. La hipótesis central, como ya se ha indicado, es que el razonamiento implica un proceso apropiado de manipulación mental de representaciones del mundo, por ello se le ha denominado **Hipótesis Representacional**, **Hipótesis del Símbolo Físico**, **Hipótesis de la Representación del Conocimiento** o **Paradigma del Procesamiento de Información** [8, 7, 2, 10].

El problema con este enfoque es que se debe contar con un modelo del mundo, una abstracción sistemática de un dominio, que ubique los conceptos relevantes y sus propiedades, para codificarlas adecuadamente. Sin embargo, esto no siempre es posible, ni mucho menos fácil. Por ello, cabe la pregunta de si resulta adecuado hacerlo o hay que buscar otros enfoques y metodologías [7, 2, 11].

Los "cerebros electrónicos" llevaron a pensar que era factible simular/emular el pensamiento humano. Se supuso que con una representación lógica, un adecuado método de resolución de problemas y una arquitectura física (como un robot, por ejemplo) se tendría un sistema inteligente. Sin embargo, esto es incorrecto para muchos de los problemas prácticos e improcedente para problemas del mundo real. Por estos motivos existen varias expectativas no cubiertas.

A modo de resumen, se menciona que la hipótesis de la representación del conocimiento (la IA Clásica) postula el siguiente panorama [2]:

a) la percepción es la construcción de representaciones internas del ambiente externo;

b) el aprendizaje es la modificación de éstas y la acumulación de nuevas representaciones;

c) la memoria es el almacenamiento y la recuperación de tales representaciones;

d) el lenguaje es la codificación, intercambio y decodificación de las representaciones;

e) el razonamiento es la manipulación lógica de las representaciones; y finalmente;

f) la toma de acciones es la ejecución de la representación de un plan a realizar.

En este orden de ideas, se menciona la metodología que impone la IA Clásica, desarrollada bajo la hipótesis representacional:

- La mayor parte de los comportamientos inteligentes pueden modelarse con ejemplares dotados de deliberación consciente.

- El razonamiento deliberativo humano es, esencialmente, una especie de computación sobre representaciones simbólicas del mundo.

- Los avances e ideas obtenidos al modelar el rendimiento de un aspecto particular de la inteligencia, restringidos a dominios particulares, se extrapolan para tener un entendimiento del comportamiento inteligente en general, para dominios no restringidos de interacción con el mundo real.

Como último comentario, hay que hacer notar que existen tareas para las cuales las computadoras son muy efectivas, pero que, en cambio, a los humanos se nos dificultan (aquellas que implican lógica y cálculos numéricos); por otro lado, hay tareas que son cotidianas y que nos parecen muy elementales, pero que resultan extremadamente difíciles o imposibles de ejecutar de manera fiable para una máquina (como el reconocimiento de rostros, de olores, caminar y subir escaleras, abrir una puerta, tomar un vaso y servir un líquido, etc.). Esto se debe principalmente a que si bien los cerebros biológicos contienden con actividades lógico-matemáticas y de abstracción, entre otras, son producto de la evolución: se desarrollaron no para hacer pruebas matemáticas, sino para la supervivencia, para controlar la conducta, resultado de un proceso de adaptación constante. De aquí parten quienes apoyan lo que se ha denominado la Nueva IA, que se expone brevemente a continuación.

2.2. La Nueva IA

Tomando en cuenta que el cerebro controla la conducta en pro de la supervivencia, algunos autores han cuestionado que la inteligencia estribe en el pensar; antes plantean que debe atenderse al comportamiento para identificarla, no como característica intrínseca, sino como una capacidad emergente. Aquí se comenta el grupo perteneciente al llamado *movimiento situado*, que apuesta por el desarrollo de agentes inmersos en el mundo real, más que en ambientes *ad hoc*,o reducidos. Cabe aclarar que un medio real no necesariamente es un ambiente natural; puede tratarse de un ambiente artificial, como el de los robots de software que habitan en internet. De esta manera, el comportamiento adaptativo, entendido como el comportamiento ajustable a las condiciones ambientales, adquiere máxima relevancia. Así, su entendimiento y síntesis se convierte en el objetivo de la IA, esto incluye el entendimiento del entorno y de las condiciones que le impone al individuo o grupo. Bajo esta nueva hipótesis, la representación es irrelevante; el ambiente es lo que moldea el comportamiento del agente; de la interacción ambiente-agente emergen los patrones adecuados [11]. A esto se le conoce como actividad situada. De acuerdo con Beer [10], la propuesta parte de la observación del entorno: todo ha sido evolución; de tal manera, hay que hacer evolucionar a los sistemas mediante su interacción con su medio real:

a) ¿Qué hacer?: revisar y replicar los mecanismos de la naturaleza y crear sistemas adaptativos al estilo de los animales.

b) ¿Cómo?: estudiando sus sistemas nerviosos.

Como se puede observar, mucho de esta actividad parte del estudio de la etología, y aun de la neuroetología, para finalmente aterrizar en la neuroetología computacional y la roboetología.

La metodología de esta nueva propuesta, descansa en los siguientes puntos:

a) la habilidad de contender de manera flexible con el mundo real es una característica que define el comportamiento inteligente, más decisiva que la deliberación consciente;

b) el comportamiento adaptativo proviene de la congruencia estructural entre las dinámicas del ambiente externo y los mecanismo internos de la entidad inteligente; y

c) para entender la dinámica requerida por el comportamiento adaptativo, es conveniente partir del entendimiento y modelado de los mecanismos neurales de control completos de animales inferiores (sencillos).

Esto conducirá, subsecuentemente, al entendimiento de la elaboración sucesiva de los mecanismos observados en los animales superiores.

De esta manera, se pretende comenzar por el entendimiento y replicación de pancompetencias animales (el conjunto completo de competencias), en lugar de buscar tener sólo microcompetencias humanas. Es decir, se propone o adopta un enfoque incremental.

2.3. La IA Hegemónica

La IA actual, ha transitado, como muchas disciplinas científicas y tecnológicas, de lo público a lo privado en cuanto a sus desarrolladores, financiadores y beneficiarios de los complejos industriales que la explotan, que están concentrados de manera mayoritaria en los países dominantes política y militarmente. Lo anterior ha moldeado de alguna manera, la forma dominante de la IA, digamos, la IA Hegemónica.

Así, el nombre de IA se ha constituido en una especie de marca, que tiende a denominar productos o servicios, y no enfoques de investigación; ni tampoco intenta referir un cuerpo de conocimientos o tecnologías. Debido al auge mediático-comercial que se vive hoy en día, el uso hegemónico del nombre IA se puede observar en dos situaciones:

- para referirse a una vastedad de tecnologías basadas en el uso de macrodatos y su explotación bajo complejos modelos estadísticos.

- en ciertas ocasiones, se le antepone un término extra con el objetivo de que se resalten ciertos rasgos o características, tal como *general*, o, como en el caso que nos ocupa, *generativa* (IAG).

Entre sus fundamentos operativos, es decir, su soporte físico-lógico, se pueden enunciar:

- el uso intensivo de datos que provienen primordialmente de la vasta miríada de redes socio-digitales y sus incontables usuarios, de las Bases de Datos recopiladas por organismos públicos y privados; de los diarios electrónicos con sus textos de noticias y artículos de opinión, de la literatura digitalizada por las grandes bibliotecas, entre otras muchas fuentes;

- la aplicación de modelos matemáticos muy robustos que utilizan el cálculo infinetesimal, el álgebra lineal y, prominentemente, la estadística;

- apoyados por vastos recursos de cómputo, como clústers y supercomputadoras (amén de numerosos centros de datos con servidores dedicados al almacenamiento y procesamiento) y poderosos recursos telemáticos (redes de área local, amplia y por supuesto internet).

En el ámbito económico, la IA hegemónica se ha convertido principalmente en un conjunto de herramientas computacionales cuyo propósito es consolidar un modelo de negocio capitalista con ciertas características, como las siguientes:

- Presenta una fuerte sumisión del aspecto técnico al económico, lo que se origina en la búsqueda de financiamiento para sostener los centros de datos, los altos salarios de ingenieros y demás especialistas de apoyo. Esta característica comercial ya estaba presente desde la conferencia de Dartmouth: tan es así que el mismo término fue pensado como un elemento de mercadeo para obtener financiamiento.

- Se aleja de lo que en un inicio se pretendía, es decir, de su misión o propósito científico: el estudio y síntesis (recreación) de entidades inteligentes. De esta forma, el énfasis está en la creación de dispositivos y sistemas de mercadeo que representen ventajas técnicas innovativas, lo que redunda en mayores márgenes de plusvalía.

- Explota la inclinación que aparentemente tenemos todos los humanos de conferir a objetos y animales propiedades humanas, es decir, de antropomorfizar objetos y otros seres. Curiosamente, esta inclinación es complementaria con otra: a la cosificación de personas, que se da bajo ciertas condiciones. Estas dos inclinaciones hacen que a los desarrollos de la IA les adjudiquemos fenómenos y características no presentes en ellos, lo que redunda en elementos de mercadeo muy potentes.

Por otro lado, la IA hegemónica, al ser la que mayor financiamiento obtiene, ha generado una amplia gama de desarrollos en muchos campos de la industria, desde el entretenimiento hasta la investigación científica, por lo que es importante analizar sus ventajas, así como permanecer críticos y propositivos ante lo que se identifica como sus desventajas.

La IA Generativa

Dentro de lo que aquí se ha denominado IA Hegemónica, se encuentra una fuerte subárea de desarrollo cuyo auge actual ha decantado todo un fenómeno mediático en torno a la IA; de hecho, se ha vuelto una suerte de sinónimo hablar de IA (en general) refiriéndose sólo a la IA Generativa (IAG).

Cuando se habla entonces de IAG, casi siempre se limita a *productos* de IAG, esto es, un complejo ensamble tecnológico dispuesto para la ejecución de tareas específicas, por ejemplo, generar escritos o imágenes.

Ejemplos de los desarrollos en esta subárea son los agentes o *bots* dialogantes o *chatbots*, traductores de idiomas, generadores de imágenes, video y código de ciertos lenguajes de programación.

De cierta manera, en el lanzamiento al público de los agentes dialogantes de la IAG, se ha sobreespeculado el poderío y los alcances de dichas tecnologías, y sobre todo se ha malentendido la forma en que debieran implementarlas en ámbitos de capital importancia, como el de la educación. De esta forma, no es vanal insistir en que aun con la demostrada utilidad que dichos desarrollos confieren a individuos y organizaciones, incluso los más publicitados[4], son sólo

[4] Como el famoso ChatGPT, Bard, Claude, entre otros.

una parte del campo de la IAG; y ésta, a su vez, se enfoca solamente en algunos aspectos de la IA, además de recalcar que ésta misma no se agota en el razonamiento automatizado; y finalmente, la inteligencia natural aun con sus dificultades y falencias, es algo mucho más amplio, rico y poderoso que todo lo anterior. Esta idea se ilustra en la figura 1.3 donde se sitúan en perspectiva la IA y sus subdisciplinas.

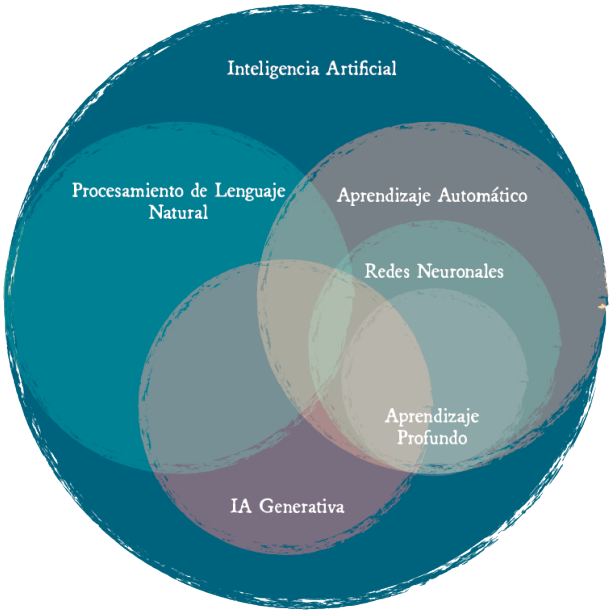


Figura 1.3: La IA es algo mucho más abarcativo que sus subdisciplinas, como el Procesamiento de Lenguaje Natural, el Aprendizaje Automático, las Redes Neuronales Artificiales, el Aprendizaje Profundo y la IA Generativa.

Dicho lo anterior, y con el afán de que el lector no se quede con una idea que pudiera desdeñar los avances e importancia de los desarrollos de IAG, a continuación se comenta sobre enfoques y tecnologías de ella, particularmente de los *chatbots*.

Generación de escritura

Estas herramientas producen textos a partir de una representación del lenguaje natural. En cierto modo, podríamos indicar que producen un *reflejo refractado* del lenguaje natural. A partir de esta representación se han creado *motores de generación de texto* que pueden sostener interacciones (comúnmente entendidas como *diálogos*, aunque no lo son) con los usuarios a partir de construcciones en lenguaje natural. Es importante decir que los *chatbots* generan *texto bien formado*, verosímil, pero no necesariamente veraz, ni correcto (en el sentido de reflejar información validada, conocimiento verdadero) [5]. Estas herramientas se ajustan, es decir, crean abstracciones estadísticas con texto generado por humanos (esto trae como consecuencia que se hereden los sesgos que en él se hallan, como el racismo, sexismo, etc., de lo que se hablará en el último capítulo

[5] Ver en el Apéndice A las consideraciones entre verdad y validez.

del libro). Por lo anterior, se recomienda no depender exclusivamente de dichas herramientas, sino echar mano de la crítica, el juicio y la creatividad humana.

Como ya se indicó, dichos motores o calculadoras de texto hacen uso de modelos de representación del lenguaje natural (ML), que son modelos de Aprendizaje Automático. En términos breves, se puede indicar lo siguiente sobre su funcionamiento:

- Dada una palabra dentro de una oración, predicen cuál será la siguiente, para lo que consideran el contexto de las n palabras anteriores.

Para explicar lo anterior, baste citar el modelo básico, que se basa en ocurrencias de n palabras o *n-gramas* en un texto. Por ejemplo:

> Buenos días a todos

podría representar los siguientes bigramas o *n-gramas* de orden 2 (*2-gramas*):

> <Buenos,días>, <días, a>, <a,todos>

A partir de estas co ocurrencias se pueden calcular estadísticas que permitirían, hasta cierto punto, estimar una secuencia de palabras dada alguna de ellas.

Los poderosos modelos actuales se basan en representaciones de complejidad superior, ya que pueden incluir miles de palabras. Para su creación se emplean mayoritariamente Redes Neuronales Artificiales de gran densidad, capaces de analizar corpus de grandes volúmenes (colecciones de texto), con los que crean una representación estadística de cómo se presentan las palabras del idioma o idiomas con los que se alimentan[6].). Una vez que aprenden esta estructura, los ML se pueden aplicar a diversas tareas, inclusive pueden reajustarse para cumplir con fines más específicos (incluso con algún corpus o corpora de un dominio en particular). Algunos de los ML más conocidos son BERT, RoBERTA (en español), o GPT-3 y superiores.

Diferencias con un buscador

Es de mucha importancia señalar que, pese a que la mayoría de las personas aún los siguen usando como buscadores, y aunque a primera vista no lo parezca, los agentes o *bots* conversacionales o *chatbots* difieren considerablemente de un buscador. Los buscadores son herramientas de consulta sobre datos indizados. Esto es, operan una vez que se ha recopilado información de un conjunto de sitios Web, textos y documentos disponibles, almacenados e indizados en internet. Entonces, su fuente de información es una sección de la web (tampoco es toda la web, como llega a suponerse), y, por ende, resulta dinámica en mayor o menor medida, dependiendo del poder de recopilación e indización del organismo que está detrás del buscador en cuestión. Así, un buscador adquiere la forma de un oráculo no dialogante, sino sólo generador de respuestas a una pregunta planteada en la ya clásica caja de texto.

Un *bot* conversacional recibe información (un fragmento de texto, presumiblemente un fragmento de un diálogo producido por un usuario o por otro *bot*), la convierte a una representación, la interpreta y da una respuesta con base en su entrenamiento previo (que, igual que el buscador, es acotado). De esta forma,

[6]Estas representaciones se denominan en inglés *transformers*

un *chatbot* no está pensado para ser un oráculo, sino que adopta la forma de un ente digital dialogante con el que se pueden establecer intercambios lingüísticos.

Es obvio que una de sus principales aplicaciones consiste en resolver preguntas en relación con un dominio específico, generalmente un servicio o producto de una organización. Sin embargo, hay que entender que no se trata de diálogos en el sentido pleno del término, y no se les debe atribuir un capacidad de respuesta adecuada para cualquier dominio, pues, como ya se dijo, generan sus respuestas; esto es, calculan secuencias de texto.

Esto último implica que las respuestas no necesariamente se apegan fielmente a la información con que se alimentaron y la que presuntamente el usuario (al emplearla para resolver dudas) está buscando. Su entrenamiento y uso debe estar circunscrito a un dominio. Así, funcionarían bien en un dominio en particular, quizá de forma más amigable con el usuario que un buscador, pero tal vez no de forma más precisa si lo que se obtiene de él son sólo líneas de diálogo en lugar de documentos o información autorizada.

Para entender el problema que se deriva del mal uso de los *chabots*, valga recordar al primero de su estirpe, denominado Eliza, el *chatbot* que provocó que su creador, Joseph Weizenbaum, se alarmara de las potenciales consecuencias de estos programas, dada la forma en que las personas humanizamos a las cosas (ya sean muñecos, objetos o sistemas de cómputo) [12]. Weizenbaum fue de los primeros en advertir sobre los dilemas éticos de ciertos usos de la IA, algo a lo que volveremos en el capítulo final de este libro.

Mucho se ha dicho y es casi una obviedad señalar que la IAG y sus productos y herramientas derivadas conforman hoy día una tecnología disruptiva. Mas ¿en qué sentido es así?

- Son disruptivas por su naturaleza (mejor dicho, por su artificialidad): Generalmente se basan en Aprendizaje Profundo (que es opaco en su explicabilidad, es decir, no se puede de forma práctica desentrañar cómo llegaron a los resultados que nos muestran)
  - Proporcionan una interacción en lenguaje natural muy avanzada.
  - Generan productos de calidad superior a los anteriores.
- Son disruptivas por nuestra propia naturaleza:
  - Aún persiste un bajo entendimiento sobre cómo emplearlas en favor de la educación y el aprendizaje.
  - Se consideran una amenaza (al menos algunos profesores lo indican).
  - Se consideran la solución (los estudiantes así lo refieren).

La IAG ofrece nuevas posibilidades en la educación para profesores y estudiantes. Para los profesionales de la educación, es factible aprovechar los *chatbots* y la IA en general para mejorar la interactividad en sus clases, en sus materiales y lecciones, también como un medio eficaz para personalizar los planes de estudio y reducir las tareas administrativas. Además, se considera útil avocarse a utilizar la tecnología para enseñar a los estudiantes a discernir entre información confiable y desinformación, con el fin de fomentar el pensamiento crítico.

Como estudiantes, la IAG puede brindarnos una experiencia de aprendizaje que infunda mayor dinamismo y con amplio margen para que sea personalizada. Los *chatbots* pueden servir como un apoyo adicional para generar interactividad

mediante conversaciones en texto natural (o multimodal) y, por ejemplo, la generación de cuestionarios adaptados a los contenidos que se ven en clase.

Es importante que estemos dispuestos a utilizar estas herramientas de manera efectiva (lo que incluye no renunciar al sentido crítico ni a la verificación de datos) y aprovechar todos sus beneficios. Se invita al lector interesado a revisar el marco de trabajo con la IAG en el contexto educativo que se propone en el apéndice C.

3. IA con enfoque de agentes inteligentes

De acuerdo con los autores Stuart J. Russell y Peter Norvig [2], la idea de *agente inteligente* puede tomarse como un enfoque para el desarrollo de la IA. De este modo, un sistema inteligente puede ser sintetizado si su diseño (y construcción) se basa en el concepto de agente, ya sea físico (robot) o de software (softbot).

Ahora bien, es claro que un robot difícilmente se consideraría sólo un producto informático, pues tiene elementos mecatrónicos que lo diferencian y especifican como algo más abarcativo; es decir, un programa de computadora con una *corporeidad*. Sin embargo, la diferencia entre un agente de software y un programa puede ser algo difusa.

3.1. Agente o programa

Para entender cuál es la diferencia entre agente y un programa de computadora cualquiera vamos a revisar algunos aspectos relacionados con ellos.

Desde luego que ambos, el agente o softbot y el programa convencional, son software, es decir programas de computadora que presentan las características que se enuncian a continuación:

- Ambos son sistemas computacionales que están insertos en un entorno que proporciona entradas (entorno-programa o agente) y recibe salidas (programa o agente-entorno).
- Ambos se realizan para cubrir una necesidad o llevar a cabo una tarea, por lo que deben cumplir con una especificación, es decir, se diseñan con un propósito definido.
- Se diseñan bajo una arquitectura y un paradigma (estructurado, orientado a objetos/eventos, funcional, entre otros).
- Comúnmente involucran una serie de algoritmos interconectados y hacen uso de estructuras de datos y estructuras de control, bibliotecas, subrutinas (procedimientos y funciones).
- Consumen recursos: a) del entorno donde operan (lo que se entiende comúmente como datos de entrada, almacenamiento externo, bases de datos, telecomunicaciones, entre otros), b) internos (sistema de archivos, memoria, tiempo de cómputo, almacenamiento interno, etc.); y con ellos proporcionan un resultado: la realización de la tarea esperada, como una serie de cálculos, un ordenamiento o un análisis de los datos de entrada.

Por otro lado, un agente o sofbot se ha definido desde diversas concepciones y posturas por diferentes investigadores, pero puede ser entendido como una entidad autónoma que mantiene interacciones con el medio ambiente donde está situado, percibiéndolo (aunque parcialmente) con sus sensores y modificándolo con sus acciones (en la medida en que es posible) en busca del cumplimiento de sus propósitos de diseño.

Al agente se le pide que exhiba ciertas características, como las que se listan a continuación:

- persistencia (perseguir sus objetivos de manera reiterada);
- autonomía (autocontrol);
- racionalidad (toma de decisiones correctas);
- habilidad social (interactuar entre sí).

Todo agente se considera *situado*; y, por ende ejecuta sus tareas explotando lo mejor posible los limitados recursos que tiene a su alcance (cognitivos, informáticos o físicos).

Para ilustrar la diferencia entre un agente y un programa convencional se muestra el esquema de la figura 1.4.

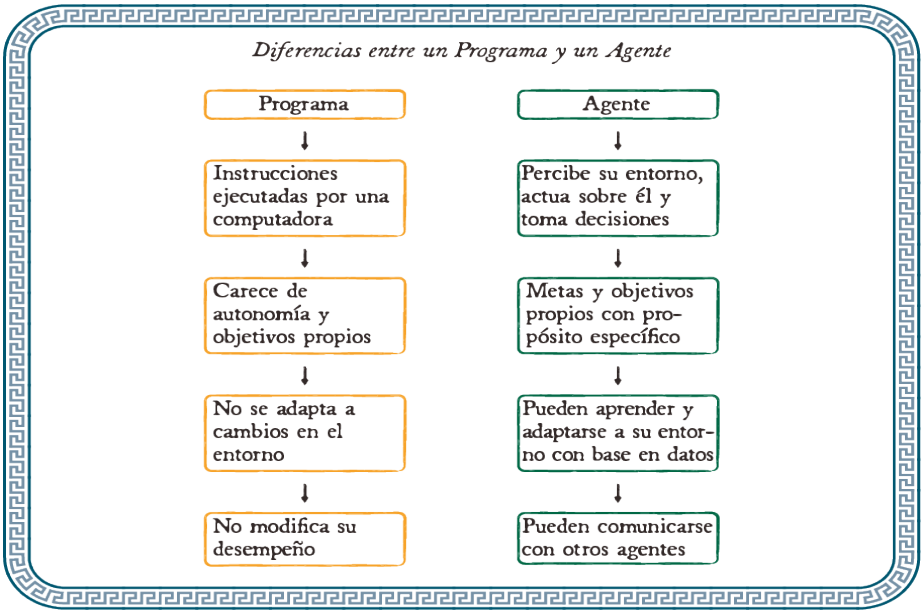


Figura 1.4: Diferencias entre Agente y Programa [13].

Ahora bien, una vez que se han destacado sus similitudes y diferencias, conviene posturlar que:

Todo agente es un programa, pero *no todo programa es un agente.*

Este libro se enfoca en proporcionar los elementos clave para entender los agentes de inteligencia artificial y dotar de ejemplos mínimos que ayuden al lector a realizar implementaciones mínimas de tales agentes con el fin de que tengas elementos que posteriormente le permitan continuar un estudio más avanzado.

4. Comentario final

La IA es el producto de varias ideas e inquietudes acerca del fascinante tema de la inteligencia, y la réplica de entidades inteligentes: nos encontramos así ante el maravilloso hecho de que seres inteligentes estudian la actividad inteligente.

Incorpora muchas ideas de diversos campos (matemáticas, filosofía, psicología, computación, etc.) y las propuestas de pensadores que se remontan hasta la época antigua de todas las sociedades del mundo.

Ahora inicia sus pasos como disciplina moderna y comienza a decantar sus metodologías y propuestas. Como industria, ha tenido varios logros, y muchas de las soluciones están aún por venir. En la figura 1.5 se pueden ver gráficamente algunos de los vaivenes de éxito y fracaso que han acontecido en su desarrollo.

Como se puede notar, la IA es una ciencia en nacimiento y formación y se perfila como un campo con mucha actividad y futuro, en el cual el enfoque de agentes parece uno de los más prometedores.

Por ello, el propósito de este texto es orientar al usuario en el entendimiento de los agentes artificiales o inteligentes, y adentrarlo en su desarrollo básico.

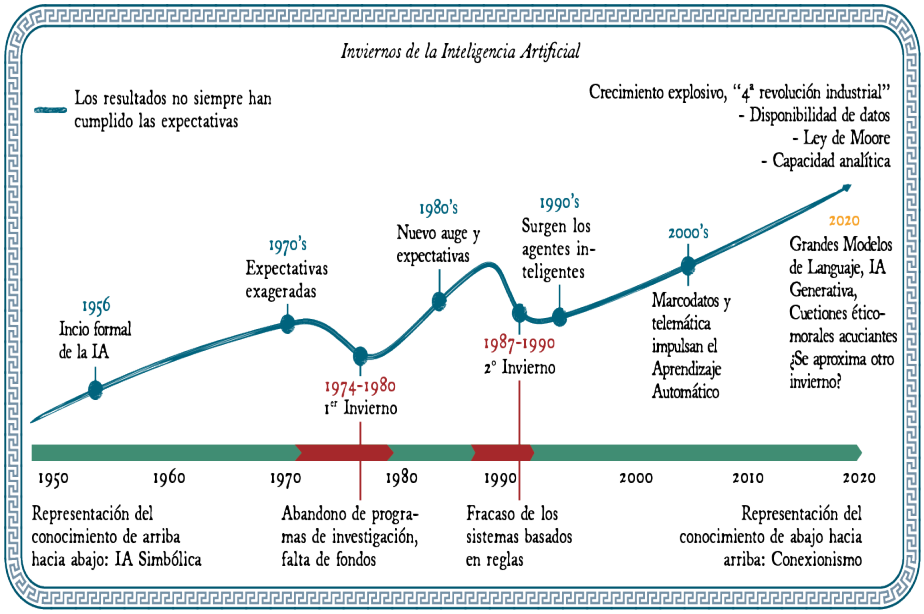


Figura 1.5: Los llamados inviernos y veranos de la IA [13].

5. Resumen

La Inteligencia Artificial

El campo de la IA es un área interdisciplinaria donde convergen conocimientos y enfoque variados.

- Su rasgo distintivo es que se orienta a la síntesis de entidades inteligentes.
- Hay varios enfoques en su desarrollo, difusamente pueden agruparse en según traten de sustentarse en un modelo de inteligencia: modelo humano, modelo de racionalidad abstracta; o por su orientación a seguir: énfasis en la acción inteligente, o en la búsqueda de un pensamiento en toda regla, pero artificial.
- A su vez, algunos de los enfoque han originado corrientes de desarrollo como la IA Clásica (postula que computar es pensar), la Nueva IA o IA Situada (busca que el mundo real sea el molde de la inteligencia de las entidades artificiales), y más recientemente la IA Hegemónica (sustentada en macrodatos, telemática y modelos estadísticos).
- La IA Generativa es una subrama con mucho auge y aplicaciones, pero no agota la totalidad de las áreas de la IA.
- Los agentes de IA son un enfoque prometedor para el desarrollo de entidades inteligentes y un concepto unificador de la disciplina.

6. Actividades de aprendizaje

6.1. Reforzamiento cognitivo

Lleve a cabo las siguientes tareas para reforzar lo aprendido

- Elabore un mapa conceptual con los conceptos clave del capítulo
- Realice una línea del tiempo con los principales logros ténicos y científicos de la IA

6.2. Cuestionario 1

Conteste las siguientes preguntas conforme la información de este capítulo [7].

1. ¿Cuál es uno de los antecedentes directos de la inteligencia artificial como ciencia moderna?

   a) La cibernética
   b) La robótica
   c) La lógica difusa
   d) Las redes neuronales

2. ¿En qué años tuvieron lugar dos hitos importantes para el surgimiento de la IA?

   a) 1940 y 1947
   b) 1950 y 1954
   c) 1957 y 1950
   d) 1960 y 1947

3. Según lo visto en el curso, los orígenes de la IA como idea pueden remontarse hasta ...

   a) La Revolución Industrial
   b) Las civilizaciones clásicas antiguas
   c) La invención de la computadora
   d) El Renacimiento europeo

4. Actualmente se observa un auge mediatico de la IA centrado en ...

   a) Logros científicos y humanistas
   b) Proyectos humanistas y de emprendedurismo
   c) Aplicaciones comerciales con macrodatos
   d) Desarrollos *open source* y aprendizaje profundo

[7] Este cuestionario fue realizado parcialmente con una IA Generativa. Ver el esquema de trabajo del Apéndice C.

5. La IA hegemónica actual se basa en:

   a) Arquitecturas neuromórficas no opacas
   b) Lógica y razonamiento simbólico profundo
   c) Técnicas de aprendizaje por refuerzo y subsunción
   d) Modelos estadísticos, aprendizaje por refuerzo y macrodatos

6. ¿Qué es la habilidad de una computadora para realizar tareas asociadas a seres inteligentes?

   a) Una definición de inteligencia artificial
   b) Un objetivo de la robótica existencial
   c) Un componente de los sistemas expertos
   d) Un tipo de aprendizaje profundo
   e) Un algoritmo de procesamiento de lenguaje natural

7. El estudio de cómo lograr que las máquinas imitien o superen a humanos en ciertas tareas se conoce como ...

   a) Aprendizaje automático
   b) Sistemas autónomos
   c) Procesamiento de imágenes
   d) Una definición de inteligencia artificial
   e) Computación evolutiva

8. ¿Qué propone una definición más integral de IA?

   a) La replicación de la conciencia humana en máquinas
   b) La creación de máquinas que piensen como humanos
   c) El desarrollo de algoritmos que imiten la cognición
   d) El estudio de la inteligencia natural por medios artificiales
   e) La evolución de sistemas cibernéticos complejos

9. En sentido extenso, la IA busca entender y replicar ...

   a) La creatividad humana
   b) La conducta inteligente
   c) Las emociones y los sentimientos
   d) El sentido común y las intuiciones
   e) Las habilidades sociales y el lenguaje
   f) Ninguna de las anteriores
   g) Todas las anteriores

10. Una meta realista actual para el desarrollo de la IA sería que las máquinas inteligentes pretendieran ...

    a) Tener conciencia propia

    b) Superar la inteligencia humana

    c) Parecerse lo más posible a humanos

    d) Realizar tareas de forma ótima y racional

    e) Desarrollar libre albedrío

    f) Ninguna de las anteriores

    g) Todas las anteriores

Capítulo 2

Agentes inteligentes

Objetivos

- Conocer el concepto de Agente Artificial y su importancia en la IA
- Caracterizar a los Agentes Artificiales
- Identificar los distintos conceptos de Agente Artificial mediante una taxonomía mínima

[...Y como tenían la apariencia de hombres, hombres fueron: hablaron, conversaron, vieron y oyeron, anduvieron, agarraban las cosas, eran hombres buenos y hermosos... Fueron dotados de inteligencia; vieron y al punto se extendió su vista, alcanzaron a ver, alcanzaron a conocer todo lo que hay en el mundo...]

Popol Vuh, tercera parte, capítulo II

1. Agentes inteligentes: un concepto integrador en IA

En su texto, fundamental para la enseñanza de la IA, Stuart J. Russell y Peter Norvig [2], proponen que el concepto de **Agente Inteligente** es suficientemente bueno para adoptarse como enfoque para el desarrollo de la IA, el cual configura una disciplina de la IA que puede extenderse como paradigma de diseño integrador de todas las subdisciplinas. Así, un sistema inteligente puede ser desarrollado (cumpliendo con la parte de *síntesis* de entidades inteligentes) como un agente, en su modalidad mecatrónica (robot) o puramente digital o de software (softbot).

Sobra decir, en refuerzo de esta idea integradora, que los robots o softbots incorporan muchos de los desarrollos de visión computacional, planificación, procesamiento del lenguaje natural, entre muchos otros.

A continuación se muestra que no sólo la IA se ha interesado por la idea de los agentes artificiales.

1.1. ¿Quién se ocupa de los agentes?

El área de Agentes Inteligentes o Artificiales (AA), y de los conglomerados que éstos pueden formar, es decir, los Sistemas Multi-Agentes (SMA), de los que se hablará más tarde, es de suyo una actividad inter y transdisciplinaria (de la misma forma que lo es la IA en su conjunto).

En la figura 2.1 se observa un esquema que ilustra las disciplinas que de alguna forma u otra aportan y utilizan los fundamentos y resultados del estudio de agentes.

Por otro lado, un hecho notable es que, en realidad, el área de AA y SMA toca un campo mayor: el de los sistemas complejos. Esto se ilustra en la figura 2.2.

Disciplinas donde los Agentes Artificiales (AA) confluyen:

- Inteligencia Artificial: estudio de la inteligencia natural por medios artificiales, enfocada a la síntesis de entidades inteligentes.
- Ingeniería de Software: se enfoca en la creación de software confiable y de calidad; emplea métodos y técnicas de ingeniería.
- Simulación Computacional: busca recrear fenómenos naturales o sociales a partir de un modelo abstracto mediante un programa informático o un sistema computacional (unitario o en red).

- Teoría de Juegos / Teoría de la Decisión: se concentran en identificar la "mejor decisión" (generalmente la que maximiza la utilidad esperada del tomador de decisiones).

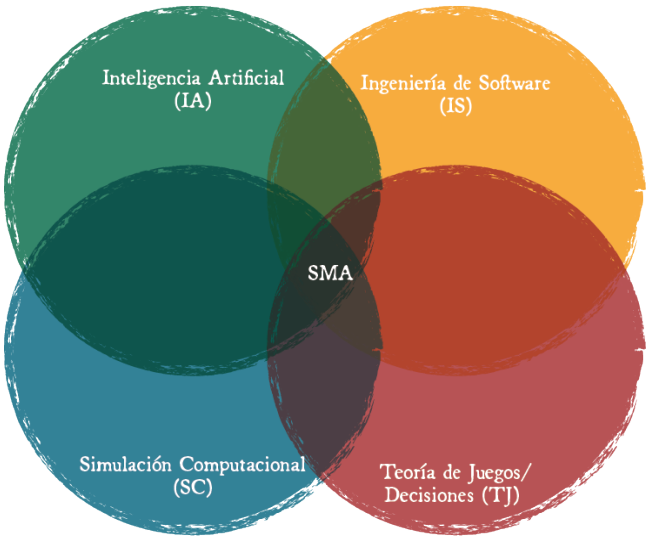


Figura 2.1: Disciplinas que convergen en el estudio de los AA y los SMA.

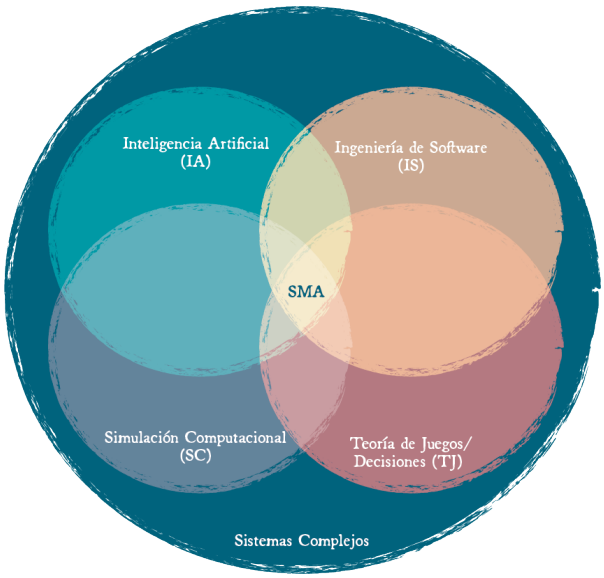


Figura 2.2: Todo se trata de lidiar con los sistemas complejos.

1.2. Concepto y antecedentes históricos

Atendiendo a su etimología y descripción, una definición simple de diccionario de la palabra agente, no es del todo prescindible, como se verá más adelante.

Algunas nociones generales de agente (del lat. *agens, -entis*, de *agere*, hacer):

- Según el *Diccionario de la Real Academia Española*[1]:
  1. adj. Que obra o tiene capacidad de obrar.
  3. m. y f. Persona que obra con poder de otra para gestionar algo en su nombre.
  6. m. Persona o cosa que produce un efecto,

A partir de estas definiciones de diccionario, podemos recapitular en la historia para seguir comparando nociones generales de agente:

- Desde la filosofía griega con Aristóteles, agente es una entidad guiada en su actuar por un propósito dentro de un contexto determinado (diría el filósofo español Ortega y Gasset "Yo soy *yo* y mi circunstancia..." [14])
- También se entiende, desde el derecho, a quien actúa en beneficio de otra persona con un propósito determinado, con responsabilidad y autoridad limitadas.

Ahora bien, de forma general, los agentes pueden agruparse en una diversidad de formas. Veamos algunas clasificaciones tomadas de la bibliografía en IA para entender un poco de la filogenia de los agentes según lo mostrado en la figura 2.3.

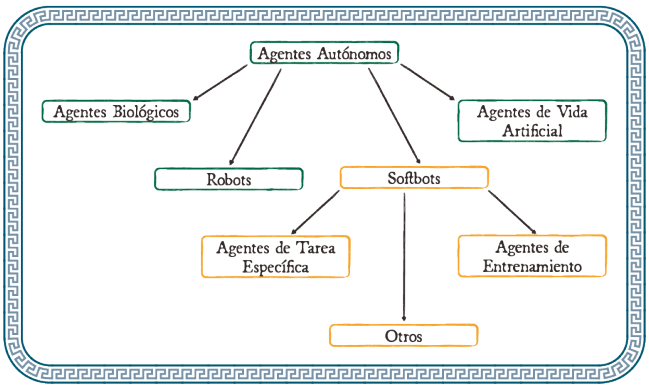


Figura 2.3: Una taxonomía de agentes autónomos. Basado en [13].

En la figura 2.4 se muestra otra taxonomía de agentes [15] según la ubicación entre ellos de los Agentes Inteligentes.

Finalmente, en la figura 2.5 observamos una tipología de Agentes Inteligentes según [16], que destaca las intersecciones entre la cooperación, el aprendizaje y la autonomía.

A partir de estas clasificaciones para aterrizar en el concepto retomamos una noción ya clásica de agente: [2].

Def. 1 *Agente: un agente es cualquier cosa capaz de percibir su* medioambiente *con la ayuda de* sensores *y actuar en ese medio utilizando* efectores.

[1] https://dle.rae.es/agente

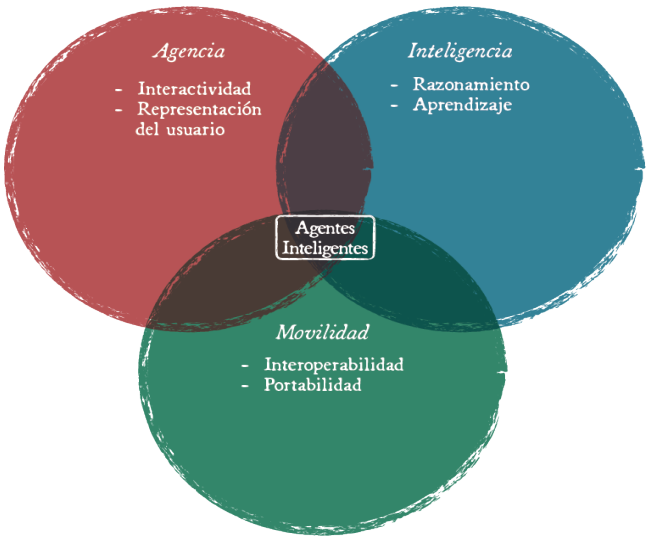


Figura 2.4: Las intersecciones que configuran los agentes inteligentes. Basado en [15].

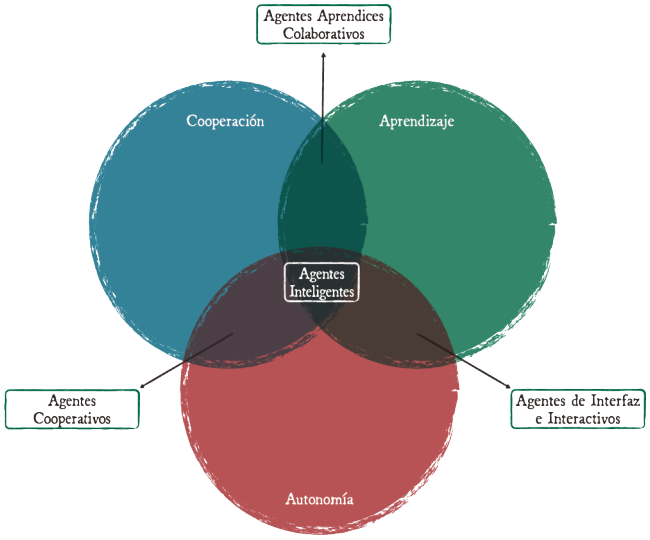


Figura 2.5: Diversos tipos de agentes, según exhiban ciertas características. Basado en [16].

Ahora bien, ¿de dónde vienen el estudio y la importancia de los agentes? Según los principales cuerpos de conocimiento de la IA, podemos identificar algunos cursos de origen:

- De la necesidad de modelar el comportamiento racional, que ofrece dos ventajas:
  - Un enfoque más general que el de las *leyes del pensamiento*: inferir correctamente es sólo una parte para obtener la racionalidad.

    - Es afín al progreso científico, que no espera una racionalidad absoluta, sino una limitada, situada en un contexto, pero parte de la primera como enfoque de largo aliento.

- De la síntesis de un ente inteligente: robots antropomorfos, zoomorfos o de distinta morfología, o bien softbots, componentes de software que actúan con tareas en un entorno específico.

- La necesidad de generar un concepto unificador de la disciplina

Por su parte, la Ingeniería de Software (IS) nos comenta que éstos devienen de las siguientes situaciones:

- La necesidad de contender con la complejidad: generar una mayor abstracción para facilitar y darle mayor eficacia al análisis.

- El modelado de software como uno o varios agentes es una metáfora natural que facilita la concepción y el diseño; permite tener sistemas abiertos que incluyan distribución de datos y control, y reutilizar el software existente.

- Como un elemento de modelado con altos niveles de abstracción. Supera los paradigmas tradicionales como los siguientes: procedural, estructurado, declarativo, orientado a objetos, patrones de diseño, *application frameworks* y *componentware*.

De igual forma, la Simulación Computacional (SC) indica algunas necesidades que orillaron al uso de agentes (particularmente, la Modelación / Simulación Basada en Agentes (MBA)):

- De la necesidad de analizar sistemas complejos: permiten realizar experimentos *in silico*. Representan estructuras y dinámicas, y recrean fenómenos asociados a la complejidad: la emergencia y la autoorganización.

- Permiten valerse de la potencia de cómputo, la disponibilidad de datos y la sofisticación que representan al modelar individuos, grupos u organizaciones como una sola entidad.

Finalmente, también la Teoría de Juegos y de la Decisión (TJ / D) aporta su punto de vista:

- Los agentes permiten describir entidades con mecanismos de decisión ingenierados *a priori*.

- Los jugadores puede modelarse como agentes para analizar sus interacciones (cooperativas o competitivas), sus mecanismos de negociación y coordinación, además de observar la relación con su medioambiente.

- Pueden ser usados para realizar negociaciones autónomas en beneficio de sus "representados" o dueños (individuos o corporaciones) dada su racionalidad y su persistencia en la consecución de su máximo beneficio.

Este libro está enfocado en las perspectivas de: IA, IS y MBA, con énfasis en la primera. De esta forma, retomamos los trabajos del área y de otras que en ella convergen. A manera de resumen, de los AA y los SMA puede decirse que:

- Tienen sus orígenes en la IA Distribuida, la Resolución Distribuida de Problemas y el Cómputo Distribuido y Paralelo.
- La Cibernética propuso un modelo análogo llamado "controlador", antecedente de los agentes.
- Los primeros trabajos sobre agentes comienzan a finales de la década de 1970 con Hewitt [17] y sus "actores" (un objeto autocontenido, interactivo y de ejecución concurrente) [18].
- Continuó en la siguiente década con varios trabajos, entre ellos los de Genesereth, Nilsson, Rosenschein y Brooks [19, 20, 21, 22, 11].
- Y tuvo su despliegue en los ochenta con los desarrollos y propuestas de investigadores como Shoham, Wooldridge, Rao, Bratman, Weiss, Etzioni, Huhns, Singh, Bordini, Hübner, entre muchos otros [23, 24, 25, 26, 27, 28, 16].

Para terminar, es necesario comentar que actualmente los AA y SMA han empezado a retomar vivacidad, sobre tanto en entornos de dispositivos móviles y de redes, pero su potencial científico y tecnológico aún está lejos de ser plenamente alcanzado.

2. Agentes Artificiales

Como se podrá ver hasta aquí, los agentes han recibido varias definiciones y concepciones, pero desde el solo hecho de ser nombrados como *inteligentes* ya tenemos un desafío entre manos. Si bien se comentó que nacen como un concepto integrador, y este adjetivo resulta adecuado, y haríamos bien en denominarlos primeramente artificiales, para después enfocarnos en caracterizar su conducta y forma de tomar decisiones para aproximarlos a una clasificación que los considere inteligentes.

2.1. Entonces, ¿qué es un agente?

Como ya se dijo en la definición 1.2, se entiende como agente a cualquier cosa (dispositivo físico o de software) que es capaz de percibir su medioambiente a través sus sensores y actuar consencuentemente mediante sus actuadores; pero hace falta profundizar en los componentes de este concepto.

En primer lugar, hay que decir que éste debería actuar *consecuentemente* con sus **objetivos de diseño**, o sea, aquello que le da fundamento a su creación como elemento tecnológico o científico.

Por *medioambiente* se entiende el entorno de trabajo donde el agente está situado, esto es, en palabras de Russell y Norvig [2], aquellos problemas para los cuales los agentes representan una solución.

Los *sensores* son aquellos dispositivos o mecanismos por los cuales el agente recibe información de su ambiente. Esta información, al ser procesada se denomina **percepción**, mediante ella, el agente se mantiene al tanto de lo que sucede a su alrededor y de cómo inciden sus acciones en el entorno.

Esta incidencia del agente sobre su medio se lleva a cabo a través de sus *actuadores*, que son dispositivos que realizan una acción en particular.

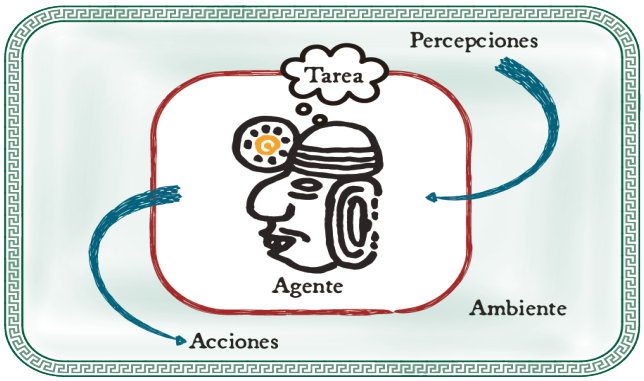


Figura 2.6: Abstracción de un AA y su interacción con el medioambiente.

En la figura 2.6 se puede observar gráficamente el concepto de agente.

Como parte de las propiedades asociadas a los agentes, dentro de la **noción débil** de agente, se encuentran la autonomía, la persistencia, la habilidad social, la reactividad, y la proactividad, que se definen a continuación:

- Autonomía: el grado de autocontrol sobre sus decisiones y acciones.
- Proactividad: comportamiento persistente orientado al logro de metas.
- Reactividad: capacidad de percibir cambios en su ambiente y poder responder a ellos.
- Sociabilididad: capacidad de interacción y comunicación con sus semejantes, otras entidades y el ambiente.

Por otro lado, tomando en cuenta la **noción fuerte** de agente, la cual postula que los agentes se especifiquen en términos aplicables a los humanos (por ejemplo usando nociones mentalistas –creencias, deseos, intenciones- o incluso se llega a hablar de agentes emocionales), éstos deben exhibir además movilidad, veracidad y *racionalidad* [23] .

Por último, la definición de agente se complementa con los siguientes conceptos:

- Medida de desempeño: define qué tan bueno es el agente, un estándar de logro de metas.
- Sensación: lo que el agente ha recibido como entrada vía sus perceptores.
- Percepción: resultado del procesamiento (interpretación) de una sensación.
- Secuencia perceptual: todo lo que el agente percibe en un lapso determinado.
- Conocimiento: datos e información sobre el medio ambiente, sobre sí mismo y otras entidades.
- Habilidad: acciones que el agente puede efectuar diestramente

- Función de agente: abstracción que determina el comportamiento de un Ag (salida) a partir de su secuencia perceptual (entrada), es la parte relacionada con su racionalidad o deliberación.

En la siguiente sección se abordarán algunos aspectos relacionados con la caracterización de agentes respecto de las características anteriormente enunciadas que dan origen a una tipología de agentes. Respecto de la racionalidad, en la siguiente subsección se abunda un poco, pero en el Capítulo 3 se abordará con mayor detalle. En relación con los agentes emocionales, en este volumen no se abordan a profundidad, sino de forma tangencial en el Capítulo 3, pero se deja el contenido para una discusión más amplia en la segunda parte de este libro.

A continuación, se abordan brevemente los conceptos clave de los agentes artificiales.

Desempeño del agente

Con el fin de aproximarse a la racionalidad, se requiere, como ya se mencionó, una manera de evaluar el comportamiento del agente. Para ello es menester establecer criterios que ayuden a determinar el éxito del agente al realizar sus tareas, tal es el objetivo de la medida de rendimiento o utilidad [2]. Sobra decir que establecer una medida de rendimiento no es una tarea fácil, pues no hay una única medida adecuada para todos los agentes. Para ayudarse en la selección de ésta, conviene seguir la regla práctica que se enuncia a continuación: diseñar medidas de rendimiento según lo que se quiere dentro del entorno, no con base en lo que se cree sobre cómo debe comportarse el agente.

Algunos ejemplos de medidas de rendimiento se muestran en la tabla 2.1.

Agente	Tarea	Medida de rendimiento
Limpiador de pisos	Aspirar un piso	Dada una unidad de superficie, maximizar el número de éstas por unidad de tiempo en un entorno dado, asociarlo con una puntuación.
Supervisor de velocidad en una avenida	Detectar a los automóviles con exceso de velocidad	Detectar autos que circulen a exceso de velocidad. Anotar un punto a favor por cada auto detectado y reportado en cierto periodo.
Tutor de música	Enseñar matemáticas de forma interactiva	Maximizar la puntuación de los educandos en las pruebas de matemáticas. Anotar un punto a favor por cada mejoría en la puntuación.

Tabla 2.1: Ejemplos de agentes y sus medidas de rendimiento.

Secuencia perceptual

El concepto de percepción indica que un agente recibe entradas de su entorno en cualquier instante (vía sus sensores). Se reitera la diferencia entre sensación y **percepción**: la primera se refiere a las lecturas que los sensores hacen del medio, es decir, los estímulos exteriores que reciben; en tanto que la percepción es la

interpretación interna que el agente hace de la sensación en términos generales, es una sensación ya procesada.

De tal manera, la secuencia perceptual SP es el historial completo de lo que el agente ha recibido hasta un momento dado [2]. Con base en ella, el agente puede tomar una decisión en un momento determinado.

Omnisciencia y presciencia

Para darnos una idea de estos conceptos, observemos el siguiente extracto del Popol Vuh:

> Las cosas ocultas por la distancia las veían todas, sin tener primero que moverse, y enseguida veían el mundo, desde el lugar donde estaban lo veían.

Éste es un caso de lo que podríamos denominar **presciencia**, es decir, cuando un agente conoce todo acerca del mundo y sus acciones presentes y futuras. Otro caso muy similar es la **omnisciencia**: un agente que tuviera esta facultad, podría estar siempre informado de todo su entorno y sabría qué resultados tendrían sus acciones (todas y cada una), por ende, actuaría escogiendo las que más beneficio le reportara. Recordemos que, dentro del concepto de racionalidad, un agente debe ejecutar las acciones que sean correctas. Por ello hay que hacer notar que, en términos reales, la omnisciencia no existe, y consecuentemente, no es posible hacer que un agente lleve a cabo siempre las mejores acciones; es decir, estamos fuera de alcanzar el modelo de un agente perfecto.

Ahora bien, algo que sí puede hacerse es que el agente se mantenga al tanto de lo que sucede a su alrededor, y para ello puede ejecutar acciones que le permitan modificar las percepciones futuras, es decir, informarse mediante una recopilación de información, o bien, a través de una exploración del entorno [2].

Finalmente, es necesario que el agente tenga incorporado un mecanismo de aprendizaje para que utilice sus percepciones como experiencia de su actuar futuro y con esto gane autonomía, que se ve reflejada cuando se apoya en sus percepciones más que en su diseño inicial. De esta manera el agente exhibe una conducta flexible que cambia los supuestos iniciales para mantenerse operativo.

Mapeo de la secuencia perceptual y las acciones

Un agente recibe estímulos del entorno y, a su vez, realiza acciones que inciden en él. Tal conversión entre percepciones y acciones es lo que se denomina *mapeo*. Un mapeo ideal constituye una especificación de qué tipo de acciones deberán emprenderse como respuesta a una determinada secuencia perceptual. En los sucesivos apartados se profundizará en la manera en que se puede realizar este mapeo percepción-acción [2].

Estructura de un agente

La IA se propone la síntesis de entidades inteligentes, tales como los agentes racionales [2]. Como ya se indicó, un agente puede ser referido a partir de su mapeo, es por ello que lo importante consiste en determinar cómo realizarlo. Partiendo de esto, un agente se concibe de la siguiente manera:

$$\text{Agente} = \text{Arquitectura} + \text{Programa}$$

Donde *arquitectura* se refiere al sustrato donde se ejecutará el programa de agente, que es la función que implanta el mapeo del agente. En cuanto a la medida de desempeño, cabe señalar que no forma parte del programa de agente. Por otro lado, la secuencia perceptual puede o no almacenarse en la memoria y manejarse de la manera más conveniente, lo que depende del entorno y de la tarea del agente.

3. Ambientes y agentes

La parte nodal de los agentes inteligentes es la especificación de su mapeo percepción-acción. Como ya se mencionó, esto corresponde al programa de agente, cuyo desarrollo es el cometido de la Inteligencia Artificial. En este punto se presentan algunos programas de agente, junto con ciertas consideraciones que hay que abordar en la síntesis de agentes inteligentes.

Una característica buscada en los agentes es la autonomía, la cual permite que éstos se adapten al ambiente, manteniéndose en operación con miras a eliminar totalmente la interacción con el diseñador. Para ello conviene abordar con detalle el tipo de tarea que el agente realizará, y sobre todo, el tipo de ambiente en el que se desenvolverá, pues éste determina la secuencia perceptual que el agente habrá de recibir. A continuación se expresan algunas consideraciones a este respecto.

Ambientes

El ambiente es el entorno donde está situado el agente. Como se dijo antes, los ambientes son aquellos problemas para los cuales los agentes constituyen una solución.

Como es evidente, existe una variedad enorme de entornos de trabajo; con el ánimo de hacer una pequeña categorización, se mencionan los siguientes tipos, en arreglo a algunas de sus características:

- Total o parcialmente observable: depende de la capacidad de los sensores para tener acceso completo o parcial al estado del medio, esto es, a los detalles relevantes para la toma de decisiones.
- Determinista o estocástico: es determinista si los estados del medio están determinados por estado anteriores y la acción ejecutada por los agentes; en otro caso, es estocástico.
- Estratégico: es el medio determinista excepto para las acciones de otros agentes.
- Episódico o secuencial: en el primer caso, la experiencia del agente ocurre en episodios atómicos (percepción + ejecución de una única acción posterior), y los episodios subsecuentes no dependen de las acciones de episodios pasados. Es secuencial cuando las decisiones actuales afectan las futuras.
- Estático o dinámico: su dinamicidad depende de si cambia mientras el agente delibera.
- Semidinámico: si el ambiente no cambia, pero sí lo hace el rendimiento del agente.

- Discreto o continuo: depende del estado del medio, del manejo del tiempo, las percepciones y las acciones del agente.
- Monoagente o multiagente: depende de la evaluación de la medida de rendimiento, si para ello influyen las acciones de un solo agente o hay ingerencia de otros.
- Competitivo o cooperativo: lo determina si las acciones de los agentes ayudan o perjudican a las de los otros.

Hay que señalar que los ambientes son muy importantes, pues determinan en gran medida la conducta y evolución de los agentes. El mundo real debe considerarse como un ambiente parcialmente observable, estocástico, secuencial, dinámico, continuo y multiagente (cooperativo y competitivo). Es, pues, el más complejo y difícil de todos los ambientes.

Tipos de agente

Los agentes se pueden clasificar de acuerdo con esta taxonomía mínima [2]:

- Agente reactivo simple (agente reflejo): estos agentes seleccionan las acciones sobre la base de las percepciones actuales, sin tomar en cuenta la secuencia perceptual histórica. Una primera aproximación a la construcción de estos agentes es el Agente Basado en una Tabla, la cual tiene un mapeo directo entre percepciones y acciones. Este primer intento adolece de rigidez (contraria a la autonomía buscada en los agentes). Además, una tabla exigiría muchos recursos en términos de diseño y búsqueda de la solución, por lo cual es un intento condenado al fracaso como solución a la mayoría de los problemas. Otro intento son los agentes que emplean reglas de condición-acción1, del tipo: *SI <condición> ENTONCES <acción>*. Su gran problema es que sólo toman la decisión correcta al analizar la percepción actual, lo cual es aplicable sólo en entornos totalmente observables.
- Agente basado en modelo: almacena información de las partes del mundo que no son visibles en el momento actual. Mantienen un estado interno para tener en cuenta la secuencia perceptual lo más completa posible. Necesita codificar la información sobre: *a)* la evolución del mundo sin considerarse él mismo (modelo del mundo); y *b)* la manera en que afectan al mundo las acciones del agente.
- Agente basado en objetivos: además del estado actual, guarda información sobre la meta que le ayude a discernir las situaciones deseables. Puede implicar un mecanismo de búsqueda y planificación para determinar la selección de acciones con el fin de lograr sus metas. Es flexible debido a que su decisión se soporta con conocimiento representado explícitamente y es modificable.
- Agente basado en utilidad: incorpora una función de utilidad que proyecta una secuencia de estados ($>= 1$). Esta función permite decidir *a)* cuando existen objetivos conflictivos y sólo puede alcanzarse uno de ellos; y *b)* cuando hay varios objetivos que guíen al agente sin la certeza de alcanzar alguno, pondera la probabilidad de éxito en función de su importancia.

Una función de utilidad explícita puede conseguir que el agente tome decisiones racionales.

Aprendizaje en Agentes

Una de las características más deseables para alcanzar la autonomía y la adaptación al ambiente es el aprendizaje y, tratándose de agentes inteligentes, se habla de **Aprendizaje Automático o de Máquina** (AM). El AM permite al agente operar en medios inicialmente desconocidos y ser más competente, al aprovechar no sólo el conocimiento inicial del diseño, sino incorporarlo del ambiente a medida que se desenvuelve en él.

Un agente que aprende consta de cuatro elementos conceptuales, enunciados a continuación [2]:

1. Elemento de aprendizaje (realiza mejoras sobre el comportamiento del agente)
2. Elemento de actuación (responsable de seleccionar acciones externas, es como el agente completo, recibe estímulos y realiza acciones)
3. Elemento crítico (hace críticas sobre la actuación del agente para influir positivamente sobre el aprendizaje)
4. Elemento generador de problemas (sugiere acciones que conducen al agente hacia experiencias novedosas y formativas, que pueden mejorar el desempeño del agente)

En el capítulo 3, sección 5 se abundará en el aprendizaje en agentes.

4. Resumen

Agentes inteligentes

Los Agentes Inteligentes Artificiales o softbots son un concepto interesante dentro y fuera de la IA, por varias razones:

- Son un concepto integrador de la IA desde el punto de vista teórico-práctico.
- Pueden categorizarse de distintas maneras siguiendo varios criterios, como el de su racionalidad, movilidad y arquitectura.
- Permiten recrear fenómenos naturales o sociales incorporando elementos de toma de decisiones de mayor o menor complejidad.
- Son unidades de comportamiento que permiten modelar y poner a prueba diversas teorías, tanto del individuo como de grupos sociales y sistemas físico-biológicos.
- Configuran una opción atractiva para el desarrollo de software de calidad.
- Tienen asociados elementos como los actuadores, y los sensores para interactuar con el medio donde están insertos.
- El ambiente que los rodea es parte relevante para su configuración y utilidad.
- Una abstracción básica de su estructura se representa por: $Agente = Arquitectura + Programa$
- Hay diversas arquitecturas y formas de implementar los programas de agente.
- Pueden incluir aprendizaje para mejorar sus comportamientos y realizar sus tareas con mayor pericia.

5. Actividades de aprendizaje

5.1. Reforzamiento cognitivo

Efectúe las siguientes tareas para reforzar lo aprendido.

- Realice un mapa conceptual con los conceptos clave del capítulo.
- Dar ejemplos de tres tareas que pueden implementarse como agentes: una para un agente reflejo, una para un agente basado en metas, y una para un agente aprendiz. Argumente la idoneidad de estos agentes para las tareas que se han indicado.

5.2. Cuestionario 2

Responda las siguientes preguntas sobre Agentes Artificiales [2].

1. ¿Qué define qué tan bueno es un agente?
   a) Medida de desempeño
   b) Conocimiento
   c) Habilidad
   d) Percepción
   e) Sensación
2. ¿Qué resulta del procesamiento e interpretación de una sensación?
   a) Conocimiento
   b) Secuencia perceptual
   c) Percepción
   d) Medida de desempeño
   e) Habilidad
3. Concepto que define las acciones que el agente puede efectuar diestramente.
   a) Sensación
   b) Medida de desempeño
   c) Conocimiento
   d) Habilidad
   e) Percepción
4. ¿Qué contiene datos e información sobre el entorno del agente?
   a) Función de agente
   b) Conocimiento
   c) Secuencia perceptual

[2] Este cuestionario fue realizado parcialmente con una IA Generativa. Ver el esquema de trabajo del apéndice C

d) Sensación

e) Percepción

5. ¿Qué determina el comportamiento del agente a partir de su entrada?

a) Habilidad

b) Sensación

c) Medida de desempeño

d) Función de agente

e) Conocimiento

Capítulo 3

Agentes Racionales

Objetivos

- Caracterizar a los agentes racionales y sus tipos básicos
- Identificar los elementos clave para programar un agente
- Programar una versión mínima de distintos tipos de agentes

¿En perseguirme, mundo, qué interesas?
¿En qué te ofendo, cuando sólo intento
poner bellezas en mi entendimiento
y no mi entendimiento en las bellezas?
Yo no estimo tesoros ni riquezas,
y así, siempre me causa más contento
poner riquezas en mi entendimiento
que no mi entendimiento en las riquezas.
Sor Juana Inés de la Cruz, *¿En perseguirme mundo qué interesas?*

1. Los agentes artificiales

En el capítulo anterior se introdujo la noción de agente (*Ag*) y se dieron algunas de sus características más relevantes. En este se revisarán más a fondo éstas y se proporcionarán los elementos mínimos para programar un Ag. La información aquí presentada se basa en gran parte en lo dispuesto por los autores Russel y Norvig [2] sobre el área de agentes.

Recordando la caracterización de un Ag, veamos a mayor profundidad el concepto de *racionalidad*, a partir de las nociones débil y fuerte de él.

Como ya se ha indicado en los capítulos anteriores, bajo la Definición 1.2, se conciben las características de autonomía, persistencia, reactividad, proactividad y habilidad social.

Respecto de la noción fuerte de agente, ésta considera propiedades que tradicionalmente se aplican sólo a los humanos:

- Usa el mecanismo de abstracción de la llamada postura intencional (*intentional stance*) de Daniel Dennett [29, 30].
- Nociones mentalistas: ayudan a modelar de manera más realista el comportamiento de seres humanos [30, 31, 32, 33]:
  - Creencias (el conocimiento sobre el ambiente y sobre sí mismo que tiene el Ag).
  - Deseos (aquello que el Ag considera que debe enfocarse en un contexto dado).
  - Intenciones (constructos que impulsan al Ag a actuar en un caso específico).
  - Emociones (variables extrínsecas y homeostáticas, funciones de recompensa, para representar e identificar emociones humanas)
  - Entre otros.

1.1. Agentes ... ¿inteligentes?

Ahora profundicemos sobre las características de los agentes antes mencionadas.

Recordando los conceptos asociados a un agente tenemos:

- Medida de desempeño/rendimiento: define qué tan bueno es el Ag, un estándar de logro de metas.

- Sensación: lo que el Ag ha recibido como entrada vía sus perceptores.
- Percepción: resultado del procesamiento (interpretación) de una sensación.
- Secuencia perceptual: todo lo que el Ag percibe en un lapso determinado.
- Conocimiento: datos e información sobre el medio ambiente, sobre sí mismo y otras entidades.
- Habilidad: acciones que el agente puede efectuar diestramente
- Función de agente: abstracción que determina el comportamiento de un Ag (salida) a partir de su secuencia perceptual (entrada), es la parte relacionada con su racionalidad o deliberación.

Consideraciones sobre *reactividad*. Toma en cuenta la interacción con el ambiente:

- Un sistema reactivo mantiene una permanente interacción con su medioambiente y responde a los cambios que ocurren en él (en un lapso adecuado para que la respuesta sea útil).

Para entender esta característica, se debe entender el *ambiente* sobre el cual opera el Ag:

- Medioambiente estático
  - El programa no necesita preocuparse sobre su éxito o falla, el programa se ejecutará ciegamente (lazo abierto), *v.gr.* un compilador.
- Medioambiente dinámico
  - Para el software es difícil construir dominios dinámicos
  - El programa debe tener en cuenta la posibilidad de fracaso

¿Y qué hay de la *proactividad*?, seguramente el lector ha escuchado y aun proferido el siguiente dicho: Tal persona (o Ag) es un x con iniciativa[1]. Bueno, veamos a qué se refiere esta iniciativa:

- Intenta generar y lograr objetivos.
- Toma iniciativa cuando es necesario.
- Reconoce las oportunidades.

En lo que respecta a la *Autonomía*, se considera lo siguiente:

- Un sistema será autónomo en la medida en que su conducta está definida por su propia *experiencia*.
- Si las acciones del agente se basan en un conocimiento integrado previamente, no es autónomo.

[1] $x = \{tonto|Ag\}$

- Un agente es más autónomo en la medida en que su comportamiento se basa:
  ⊕ en el aprendizaje, y
  ⊖ en el conocimiento incorporado.

Este concepto trasciende la computación y se retroalimenta de filosofía y las ciencias políticas y cognitivas:

- El *Ag* actúa basándose en sus metas.
- Debe poder seleccionar cuál meta procesar en cada momento
- Su existencia trasciende el tiempo estricto requerido para cumplir sus metas.
- Es robusto y se mantiene viable pese a cambios ambientales.
- Procesa información del ambiente mediante sus interacciones con éste.
- Su comportamiento es fluido y adaptable: con una variedad de respuestas.
- Es selectivo en cuanto a los estímulos que atiende al momento.
- No es un títere o algo a control remoto: ninguna entidad externa lo manipula.
- Una vez que inicia su operación, no requiere ser reprogramado.

En cuanto a una característica clave, la *Sociabilidad*, podemos decir que el mundo real es colectivo, no se puede ignorar a los demás: se requiere habilidad social. La sociabilidad entonces puede caracterizarse como sigue:

- Es la habilidad para *interactuar* con otros agentes (posiblemente humanos)
- Se vale del uso de ciertos lenguajes/protocolos de *comunicación* entre agentes.
- Busca la *cooperación*, pues ciertas metas pueden lograrse únicamente en conjunto con sus semejantes.
- Esto es lo que permite crear ensambles de agentes: Multi-Agentes.

Una característica a menudo dada por sentada en los agentes es la de la *persistencia*, pero es necesario explicarla un poco. Expresa la continuidad en el comportamiento de un agente:

- Un Ag está ejecutando continuamente su ciclo de procesamiento: $percibir - decidir - incidir$.
- Esto es, su funcionamiento orientado al logro de sus propósitos es continuo, persistente.

Respecto del *Aprendizaje*, ya se había indicado previamente sobre lo interesante que resulta que un agente sea capaz de usar algoritmo para aprender de su propia experiencia, esto es:

- El Ag trata de aprender para mejorar su rendimiento en el tiempo.

- Se trata de crear Ag capaces de *generalizar comportamientos* a partir de una información no estructurada suministrada en forma de ejemplos.
- Se puede entonces aprender vía un proceso de inducción del conocimiento, entre muchos otros mecanismos.

Ahora bien, para conseguir desarrollar agentes inteligentes, lo mejor es operar por pasos incrementales que nos acerquen más y más a la noción de inteligencia. Uno de estos pasos es la operativización de la tan preciada *racionalidad* (es decir, la disposición de medios algorítmicos para implementar esta noción). A continuación se ahonda en esta característica y los medios para conseguir agentes racionales (programarlos).

1.2. Hacia un agente inteligente: agentes racionales

El principio por el cual son concebibles los agentes inteligentes es el de *racionalidad*, la cual guía la conducta hacia el accionar de forma correcta, entendiendo por correcto aquello que le permite al agente alcanzar mejores resultados (en sus tareas) tomando en cuenta sus capacidades (siempre limitadas), en un momento determinado.

Lo relevante de una **Conducta Racional** es hacer lo que se debe hacer (es decir, lo *correcto*).

Pero ¿qué es lo racional para un Ag? Se dice que un agente es racional si hace lo correcto (diría P. Maes: *do-the-right-thing* [34])

- Operacionalizar la racionalidad: una *acción correcta* es la que lleva al Ag a tener éxito ⇒ definir *cómo* y *cuándo* se considera que un Ag tiene éxito.
- Recordemos la *medida de rendimiento*: medida de éxito en una tarea determinada.
- El éxito es siempre un *éxito esperado*, es decir, dependiente de lo percibido, sus recursos y habilidades en un entorno dado para una tarea definida.

La racionalidad en agentes estriba en cuatro factores básicos, que nos permiten operacionalizarla (disponer un conjunto de funciones, métricas, algoritmos para su tratamiento computacional):

1. Las acciones que el Ag puede efectuar.
2. El conocimiento (C) acumulado del medio (*Amb*) donde está situado.
3. La *secuencia perceptual* (SP) del Ag hasta un momento determinado.
4. La *función de rendimiento* (FR) que define el criterio de éxito con el que el Ag realiza sus acciones.

De lo anterior se deriva la siguiente definición de agente racional, basada en lo expresado por Russell y Norvig [2]:

Def. 2 ***Agente Racional*** *(Ag_R):* *Dada cualquier SP, un Ag_R deberá realizar aquellas acciones A que supuestamente maximicen su medida de rendimiento MR, con base en las evidencias aportadas por la SP y el conocimiento C que el agente mantiene almacenado.*

Consideraciones sobre la MR (para definir posteriormente una FR):

- Evalúa el *cómo*.
- Su pregunta base es ¿qué tan exitoso ha sido un Ag? (tras una acción o serie de ellas).
- Debe ser objetiva.
- Se debe diseñar medidas de rendimiento según lo que se quiere dentro del entorno, no con base en lo que se cree sobre cómo debe comportarse el Ag.

Ahora bien, debemos revisar el concepto de ambiente de forma más detallada.

2. Ambientes

Como se ha indicado desde el inicio de este texto, el ambiente es el escenario real o artificial donde está inserto el agente, desde donde recibe información y en donde se deja sentir el efecto de sus acciones. A continuación se hablará con un poco de más detalle sobre los ambientes referidos a los agentes.

2.1. Noción de Ambiente

El ambiente (*Amb*) o entorno de trabajo donde operan los agentes se puede pensar desde varias nociones.

- El *Amb* es el *problema* para el que los Ag_R son la *solución*.
- Es el espacio donde un agente o un grupo de ellos se encuentran situados.
- Se señala que deben procurarse *Amb* del mundo real:
  - *Amb* físicos (a la David Brooks) $\Rightarrow$ robots [11].
  - *Amb* virtuales (a la Oren Etzioni) $\Rightarrow$ softbots [27].
- Lo relevante será que la interacción $Ag_R - Amb$ mantengan la definición de Ag_R: con persistencia temporal y autonomía.

Para definir un FR, se debe entender el entorno o ambiente y su relación con el Ag_R

- La relación $Ag_R - Amb$ es siempre la misma: el Ag_R ejerce acciones sobre el *Amb*, que a su vez aporta percepciones al primero.
- Es necesario considerar además que un Ag_R siempre está *situado*:
  Un Ag_R siempre es parte de un *Amb*; percibe el entorno, y actúa con base en ello.

Incluso, un Ag_R se puede concebir desde la interacción que sostiene con su *Amb*

- Un Ag_R realiza 3 funciones de manera continua:
  - Percibir de manera dinámica las condiciones del *Amb*.

  - Razonar para interpretar las percepciones; resolver problemas; hacer inferencias y determinar acciones a tomar.
  - Actúa para afectar las condiciones del *Amb*.

- El programa de agente podría resumirse como el bucle $percibir - decidir - incidir$.

2.2. Tipos de Ambientes

Russell y Norvig plantean una tipología ya clásica de *Amb* [2].

Ambiente	**Descripción**
Accesible / inaccesible	Accesible, si los sensores detectan los aspectos que requiere el *Ag* para elegir una acción.
Deterministas / estocásticos	Determinista, si el estado siguiente de un ambiente se puede determinar completamente con el estado actual y las acciones escogidas por el *Ag*, en otro, caso es estocástico.
Episódicos / secuencial	Episódico, cuando la experiencia del *Ag* se divide en episodios atómicos (percepción + ejecución de una única acción), y los episodios subsecuentes no dependen de las acciones de los anteriores. Secuencial, cuando las decisiones actuales afectan a las futuras. Si es episódico, es más simple.
Estáticos / dinámicos	Estático, si el medio ambiente no cambia mientras el *Ag* se encuentra deliberando.
Semidinámico	Si el ambiente no cambia, pero sí lo hace el rendimiento del *Ag*.
Discreto/ continuo	Depende del estado del medio, del manejo del tiempo, las percepciones y las acciones del *Ag*. Discreto, si existe una cantidad limitada de percepciones y acciones distintas y distinguibles.
Observable total / parcialmente	Depende de la capacidad de los sensores para tener acceso completo o parcial al estado del medio, esto es, a los detalles relevantes para la toma de decisiones.
Estratégico	Es el medio determinista excepto para las acciones de otros *Ag*.
Monoagente / multiagente	Depende de la evaluación de la medida de rendimiento, si para ello influyen las acciones de un solo *Ag* o hay injerencia de otros.
Competitivo / cooperativo	Lo que lo determina es si la acciones de los *Ag* ayudan o perjudican a las de los otros.

2.3. Entornos de Trabajo

Diseñar un Ag_R es una actividad que pasa por especificar el **entorno de trabajo** lo más completo posible vía el RAES / REAS, siglas usadas para los siguientes conceptos:

- **R**endimiento

- **A**mbiente o **E**ntorno
- **E**fectores o **A**ctuadores
- **S**ensores

Ejemplo 1 de RAES. Un Ag_R para un taxista robótico/vehículo autónomo

$\mathbf{Ag_R}$	R	A	E	S
Taxista	Seguridad, rapidez, legalidad, confortable, máximo beneficio	Carretera, tráfico, peatones, clientes, competidores	Dirección, acelerador, freno, luces y señales, bocina, visualizadores y dispositivos de salida	Cámaras, GPS, sonar, velocímetros, tacómetros, infrarrojos, comunicaciones y dispositivos de entrada

Ejemplo 2 de RAES. Dado el *Ag* `xbiff`, un demonio del sistema `X Windows` en ambientes GNU-Linux/Unix

Ag	R	A	E	S
`xbiff`	Rapidez del aviso de nuevo correo, identificar el buzón del usuario correcto	Sistema de archivos Unix y su sistema de tuberías (*pipeline*)	Redireccionamiento de salida, interfaz gráfica, escritura de archivos	Lectura de archivos, re direccionamiento de entrada, teclado, ratón, otros

NB: nótese que `xbiff` no se considera un Ag inteligente, quizá podría ser sólo un Ag_R.

Dado el *AgR Tutor Inteligente de Música (TIM)*, situado en ambientes GNU-Linux/Unix, completar el RAES

Ag	R	A	E	S
`TIM`	...	...	...	...

Respuesta: Dado el *AgR Tutor Inteligente de Música (TIM)*, situado en ambientes GNU-Linux/Unix, completar el RAES

Ag	R	A	E	S
`TIM`	Maximizar la puntuación de los alumnos en las evaluaciones	Sistema de archivos Unix y su sistema de tuberías (*pipeline*)	interfaz multimodal para desplegar contenidos, ejercicios, sugerencias, correcciones, evaluaciones	interfaz multimodal, teclado, ratón, otros

3. Programas de Agente

En esta sección se introducen los elementos básicos para implementar agentes artificiales basándose en una clasificación general: reactivos y cognitivos. En ambos casos cabe señalar que se pueden implementar en cualquier lenguaje de programación, eso depende de la aplicación concreta a la que se orienten, por lo que en este libro se toma como base programas escritos en el lenguaje Python para los primeros, que por su popularidad conviene como lenguaje ilustrativo.

3.1. Implementar un agente racional

Para iniciar, conozcamos los tipos de agente racional (Ag_R), pues existen varias formas de implementar un Ag_R y muchos enfoques:

- Agentes Reactivos Ag_r
  - Reflejo Simple (es dubitable su carácter de Ag)
    - Ag_{rt} Basado en una Tabla
    - Ag_{rf} Basado en una Función o Algoritmo
    - Ag_{rp} Basado en Producciones
  - Ag_{mod} Reactivo Basado en Modelos
  - Ag_{obj} Reactivo Basado en Objetivos
  - Ag_{utl} Reactivo Basado en Utilidad
- Agentes Cognitivos Ag_C
  - Ag_{BDI} Agente BDI
- Ag_{Xapr} Agente de un tipo X extendido con aprendizaje
- Entre muchos otros...

Comencemos por la programación de agentes racionales reactivos, ya que los agentes cognitivos se cubrirán en la segunda parte de esta obra.

Entonces, ¿cómo programo un Ag_R? Si se recuerda el postulado de que un Ag tiene codificadas sus posibilidades de acción dentro de un *programa*, esto es, el conjunto de algoritmos y ténicas que le permitirán responder al ambiente, implementar su forma de realizar la selección de sus acciones en función de lo que percibe de su ambiente. El problema consiste entonces en definir una arquitectura y en implementar un programa de Ag que opere sobre esta. En otras palabras, sintetizar un Ag_R implica diseñar su **programa de agente** (P_{Ag}), lo cual considera lo siguiente:

- Comportamiento: lo definen las acciones (A) en relación a su SP.
- Más allá de su comportamiento, el P_{Ag} determina su diseño interno:
  - Su **función de agente** (F_{Ag}) que convierte percepciones en acciones $\Rightarrow$ hace un *mapeo*.
  - **Mapeo Ideal**: especificación sobre qué acciones deberían tomarse frente a una SP dada.

Hay que considerar entonces una *Arquitectura de agente*. Esto es, definir al agente en términos de una arquitectura dada, recordemos: $Ag = Arquitectura + Programa$

- **Arquitectura de agente** (Arq_{Ag}): sensores-procesamiento-efectores
- $Ag = Arq_{Ag} + P_{Ag}$
- P_{Ag}: implementa la F_{Ag} que realiza el *mapeo*

Cabe señalar que la SP puede almacenarse o no, según convenga. En tanto que la MR / FR no forma parte del P_{Ag}, se establece de manera separada.

¿Cómo programo una F_{Ag} para un Ag_r? Para implementar la F_{Ag} considérese lo siguiente:

- $Ag_r : E \rightarrow A$
  Donde:
  A: conjunto de acciones disponibles para el Ag_r
  E: conjunto de estados en el ambiente (interpretado vía sus percepciones)

El algoritmo base de los P_{Ag} se puede ver en el listado 3.1:

```
Función agenteReflejoSimple(Percepción p): Acción
  Var Acción a
  % realiza el mapeo percepción-acción
  a <- interpretar(p)
  responde(a)
```

3.2. Agentes Reflejos Simples

Agente Reflejo Simple Basado en una Tabla

El Agente Reflejo Simple Basado en una Tabla (Ag_{rt}), a veces conocido también como **Autómata** o **Agente Tropista** puede definirse breve y (se espera) lúdicamente como sigue:

- Es el "hola, mundo" de los agentes.
- Testarudo, a la menor provocación actúa acorde con sus mandamientos, escritos en tablas de relación `percepción/acción`.
- Carece de una representación simbólica explícita del mundo.
- Olvida su historia y es austero: requiere poco poder de cómputo. Selecciona las acciones sobre la base de las percepciones actuales, sin tomar en cuenta la SP:
  - Ag_{rt}: la tabla tiene un *mapeo directo* entre percepciones y acciones, en ese sentido es un mapeo ideal.
- Su comportamiento inteligente emerge de las interacciones con otros y con su medio.

En suma, es el Ag_{rt} es el primer intento de construcción de los Agentes Reflejos Simples. Ahora bien ¿Cómo programo una F_{Ag} para un Ag_{rt}?. Para ello, hay que considerar el algoritmo base de los Ag_{rt} mostrado en el listado 3.2:

```
Función agenteRSBasadoEnTabla(Percepción p): Acción
  Var:
  %Acción, inicializada con nulo/no acción
  Acción a
  %Tabla indizada de mapeo percepción-acción
  ListaAcción T = {Percepción-Acción}
  %Busca en T el mapeo que corresponda según p
  a <- consulta(p,T)
  responde(a)
```

Ahora bien, revisemos algunos problemas del Ag_{rt}:

- Adolece de rigidez (contra la buscada autonomía de un Ag)
- Una tabla exigiría muchos recursos en términos de diseño y búsqueda de la solución.
- Intento condenado al fracaso como solución a la mayoría de los problemas reales.
- Sólo toman la decisión correcta al analizar la percepción actual, lo cual es aplicable sólo en entornos totalmente observables.

Agente Reflejo Simple Basado en una Función

Ante la situación precaria del Ag_{rt}, se presenta el **Agente Reflejo Simple Basado en una Función** (Ag_{rf}), que es un primer intento de mejora del anterior:

- Incorpora una función o algoritmo en sustitución de la tabla:
  - Ag_{rf}: la función/algoritmo realiza el *mapeo directo* entre percepciones y acciones.
  - Selecciona las acciones sobre la base de las percepciones actuales, sin tomar en cuenta la SP.

Sin embargo, este primer intento también es dubitable en cuanto a su carácter de Ag_R. ¿Ideas de por qué ocurre esto?

Mientras lo reflexiona el lector, veamos en el listado 3.2 cómo programar una F_{Ag} para un Ag_{rf}.

Algoritmo base de los Ag_{rf}:

```
Función agenteRSBasadoEnFunción(Percepción p): Acción
  Var:
  %Acción, inicializada con nulo/no acción
  Acción a
  %Aplica un algoritmo de mapeo percepción-acción
  a <- mapeo(p,T)
  responde(a)
```

Agente Basado en Reglas

Otra mejora es la representa el **Agente Reflejo Simple Basado en Reglas o Producciones** (Ag_{rp}) que es la primera aproximación seria de Agentes Reflejos Simples:

- Utiliza **reglas de condición-acción** también llamadas **producciones**.
- Producción: regla `SI<CONDICIÓN>-ENTONCES<ACCIÓN>`.
  - (Ag_{rP}): el *mapeo* entre percepciones y acciones se da a través las conjunciones de disyunciones $((\alpha \wedge \beta) \vee (\alpha \wedge \beta))$ o disyunciones de conjunciones $((\alpha \vee \beta) \wedge (\alpha \vee \beta))$ representadas en las producciones.

En la figura 3.1 se muestra gráficamente este tipo de agente.

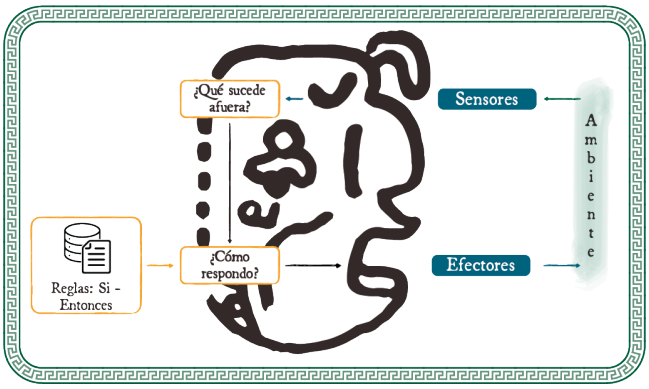


Figura 3.1: Agente Reactivo Simple Basado en Reglas o Producciones.

Sin embargo, también hay que resaltar su gran problema: selecciona las acciones sobre la base de las percepciones actuales, sin tomar en cuenta la SP (aplicable sólo en Amb totalmente observables)

¿Cómo programo una F_{Ag} para un Ag_{rp}? Veamos el algoritmo base de los Ag_{rp} en el listado 3.2:

```
Función agenteRSBasadoEnProducciones(Percepción p): Acción
 Var:
% Acción, inicializada con nulo/no acción
  Acción a
% Una producción o regla SI<>ENTONCES<>
  Regla r
% Conjunto de producciones
  R = {reglas de condición-acción}
% Busca la regla aplicable a p y la ejecuta
  r <- reglaAplicable(p,R)
  a <- aplicaRegla(r)
  responde(a)
```

3.3. Agente Basado en Modelo

Agente Basado en Modelo o Agentes con estado interno:

- Celoso del acontecer cotidiano; conoce su historia y guarda una representación del mundo y sus cambios (acciones/consecuencias).

Agente con Estado Interno (Ag_{mod}) o Agente Basado en Modelo:

- Almacena información de las partes del mundo que no son visibles en el momento actual, mantienen un *estado interno* para tener en cuenta la SP lo más completa posible
- Necesita codificar la información sobre:
  - La evolución del mundo sin considerarse él mismo (modelo del mundo), y
  - la manera en que afectan al mundo las acciones ya tomadas.

En la figura 3.2 se puede apreciar este tipo de agente.

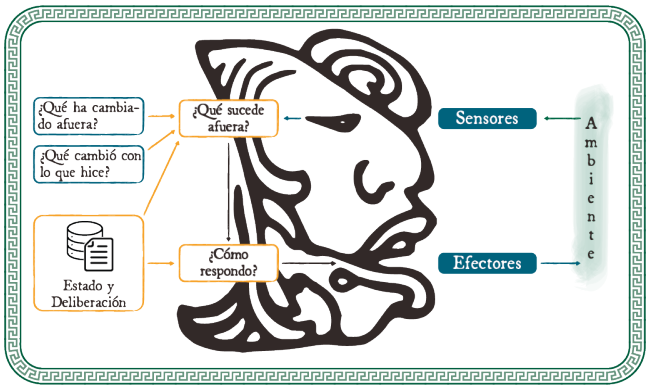


Figura 3.2: Agente Basado en Modelo (con estado interno).

¿Cómo programo una F_{Ag} para un Ag_{mod}? Algoritmo base de los Ag_{mod}:

```
Función agRSBasadoEnModelo(Percepción p): Acción
  Var:
% Estado del mundo y Estado Interno
  EstadoMundo em,   Estado e
  Regla r, R = {reglas de condición-acción}
% Acción inmediata anterior (nula al inicio)
  Acción a
% Interpreta el edo del mundo em y actualiza su edo interno e
  em <- determinarEdoMundo(p,a)
  e <- actualizarEdoInterno(em)
% Busca la regla aplicable al edo interno e y la ejecuta
  r <- reglaAplicable(e,R)
  a <- aplicarRegla(r)
  responde(a)
```

3.4. Agente Basado en Objetivos

En cualquier escenario de operación se plantea a los agentes la ejecución de múltiples tareas, cada una de las cuales se considera una meta a alcanzar. De lo que surge la pregunta ¿Cómo alcanzar una meta específica? Es en este orden de ideas, en el presente apartado atenderemos específicamente a un tipo de agente basado en objetivos o metas: el **Agente que Soluciona Problemas**. Este agente nace como respuesta ante la necesidad de alcanzar una meta a través de una secuencia de acciones. Como se plantea en la Parte 3, el mapeo del agente es el elemento crucial para definir su comportamiento (y es el cometido de su construcción).y realizar este mapeo no es una tarea trivial. Por ello, en un intento por simplificar el diseño, se plantea el logro de metas como objetivo del agente.

Los agentes que resuelven o solucionan problemas, son esencialmente **Agentes Basados en Objetivos** (Ag_{obj}) y se caracerizan como sigue:

- Toma en cuenta el futuro: además del estado actual, guarda información sobre las acciones que le ayuda a discernir las situaciones deseables para alcanzar sus objetivos.
- Puede implicar mecanismos de *búsqueda* y *planificación*[2] para determinar la selección de acciones con el fin de lograr el objetivo propuesto.
- Es flexible, debido a que su decisión se fundamenta con conocimiento representado explícitamente y es modificable.
- Implica una mejora respecto de los Ag_r: en tanto que no es puramente reflejo, implica cierta *deliberación* (buscar las acciones a realizar para cumplir el objetivo).

En la figura 3.3, se muestra gráficamente el agente Basado en Objetivos.

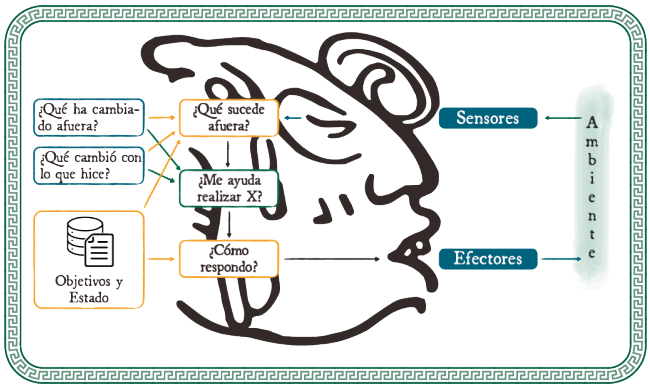


Figura 3.3: Agente Basado en Objetivos (que resuelve problemas)

¿Cómo programo una F_{Ag} para un Ag_{obj}?

Para implementar estos agentes, hay que recurrir al algoritmo base de los Ag_{obj}:

[2] Dos grandes áreas de investigación en IA.

```
Función agBasadoEnObjetivos(Percepción p): Acción
  Var:
% Lista de acciones (inicialmente vacía)
  ListaAcción SA = {}
  EstadoMundo em, Acción a
% Especificación de un problema (inicialmente nulo):
  Estado e, Objetivo o, Problema pr = {}
% Interpreta sus percepciones y actualiza su estado interno
  em <- determinarEdoMundo(p,a)
  e <- actualizarEdoInterno(em)
% Deliberación: si no hay acciones, generar un objetivo y su lista de acciones
%               luego, ejecutarlas hasta conseguir el objetivo
  si(SA == {}) entonces:
    o <- formularObjetivo(e)
    pr <- formularProblema(e,o)
    SA <- busqueda(pr)
  a <- primera(SA)
  SA <- resto(SA)
  responde(a)
```

Consideraciones sobre los Ag_{obj}

- A los Ag se les plantea la ejecución de múltiples tareas, cada una de las cuales se considera una meta a alcanzar.
- Este Ag es una respuesta ante la necesidad de alcanzar una **meta** a través de una secuencia de acciones.
- El diseño se simplifica si se plantea el logro de metas como objetivo del Ag_{obj}.
- De lo que surge la pregunta ¿cómo alcanzar una meta específica?

Para complementar el diseño de un Agente Basado en Objetivos (Ag_{obj}), es necesario analizar el tema de búsqueda en un espacio de estados, sin embargo, como ese tema es muy amplio, se destina para el capítuo 4, por ahora lo que podemos decir solamente es que el diseño de un agente que soluciona problemas se puede ver como la repetición de las siguientes actividades:

> {Formular metas, formular problemas para lograr cada meta, buscar la solución, ejecutarla}

3.5. Agente Basado en Utilidad

Una vez que se ha conseguido que un agente mantenga uno o varios objetivos, surge la cuestión de que, en determinados momentos, el ambiente le presente un desafío al configurar una situación donde se tenga que optar por encaminarse a alcanzar un objetivo y no otro, al menos dentro de una ventana de tiempo determinada.

Así, surgen los agentes que resuelven conflictos entre objetivos (Ag_{utl}), entendidos también como Agente Basado en Utilidad (discierne entre objetivos en conflicto), que se caracterizan como sigue:

- Incorpora una función de utilidad FU, que proyecta una secuencia de estados a números reales ($\geqslant 1$).
- La *FU* permite decidir:
  - Cuando hay objetivos conflictivos y sólo puede alcanzarse uno de ellos, y
  - Cuando hay varios objetivos que guíen al Ag_{utl} y no hay certeza de alcanzar ninguno: pondera la probabilidad de éxito en función de su importancia.
- Una FU explícita puede hacer que el Ag_{utl} tome decisiones racionales (las que proporcionan mayor rendimiento).
- Se puede incorporar un algoritmo de propósito general no dependiente de la FU.
- Ergo, la racionalidad como definición global se convierte en una restricción local de diseño

En la figura 3.4 se puede apreciar el Agente Basado en Utilidad.

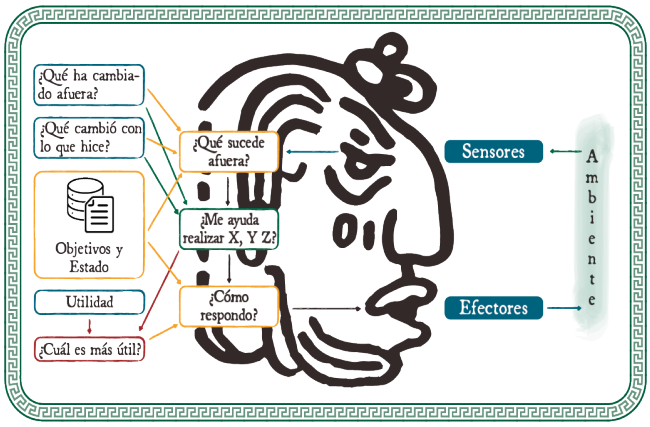


Figura 3.4: Agente Basado en Utilidad (discierne entre metas en conflicto)

¿Cómo programo una F_{Ag} para un Ag_{utl}? Algoritmo base de los Ag_{utl}:

```
Función agBasadoEnUtilidad(Percepción p): Acción
Var:
% Lista de Utilidades por acción (números reales >=1)
  ListaUtilidad LUA = {}
  ListaAcción SA = {}
  EstadoMundo em, Acción a
  Estado e, Objetivo o, Problema pr = {}
  em <- determinarEdoMundo(p,a)
  e <- actualizarEdoInterno(em)
% Deliberación: discierne entre metas en conflicto
  si(SA == {}) entonces:
    o <- formularObjetivo(e)
```

```
    pr <- formularProblema(e,o)
    SA <- busqueda(pr)
% Aplica la FU para considerar las consecuencias para cada acción
  LUA <- FU(SA)
% Toma la acción que maximice la utilidad
  a <- max(LUA)
  SA <- SA - a
  responde(a)
```

4. Agentes Aprendices

Esta sección aborda una de las propiedades comúnmente asociadas con la inteligencia: el aprendizaje, que, como se ha dicho, está identificado claramente con la acumulación de experiencia y la mejora en la realización de tareas.

4.1. Agentes que aprenden

Los agentes que tienen la capacidad de aprender, o Agentes Aprendices, pueden sintetizarse, informalmente, en unas pocas líneas:

- Tocado por Prometeo: aguza sus habilidades a través de la experiencia para efectuar sus tareas de mejor manera.
- Aprende extrayendo conocimiento del ambiente.

De esta forma, los agentes aprendices (Ag_{apr}) se caracterizan por aprender de su experiencia, es decir, de extraer de su interacción con el ambiente los datos y la infomación necesarias para adquirir habilidades o mejorar la forma en que realizan algunas de sus actividades o tareas. Así, un agente que aprende de su experiencia despierta la discusión sobre ciertas cuestiones:

- Una de las características más deseables para alcanzar la autonomía y la adaptación al ambiente, es el *aprendizaje*.
- Tratándose de Ag, se habla de Aprendizaje Automático o de Máquina (AA) como una herramienta que los dota de esta capacidad.
- El AA permite al agente operar en medios inicialmente desconocidos y ser más competente, al aprovechar no sólo el conocimiento inicial del diseño, sino incorporarlo del ambiente a medida que se desenvuelve en él.
- El aprendizaje es un proceso de mejora vía la experiencia del Ag al realizar una tarea
- ¿Qué aprender?
  - Cómo seleccionar acciones.
  - En el contexto de los SMA: cómo otros toman acciones, cuáles son sus metas, planes, creencias...
- ¿Cuándo aprender?

- En dominios complejos que dificultan una solución directa (diseñada) o cuya planificación automática es intratable, el dominio es poco conocido o altamente cambiante.

4.2. Aprendizaje de Máquina

Esta característica de los agentes es la que más se asocia con un rasgo clave de la inteligencia: la mejora de la ejecución y la adquisición de nuevas habilidades. En el contexto de los agentes tenemos que referirnos al Aprendizaje de Máquina o Aprendizaje Automático, una rama de la computación que se especializa en análisis de datos para extraer de ellos una representación operativa, generalmente muy suscinta en comparación con los datos para que pueda ser utilizada posteriormente para los fines de diseño del agente (sin tener que referirse a los datos).

En otras palabras, se trata de desarrollar programas de computadora que:

- adapten su comportamiento automáticamente (conforme datos observados)
- eviten la necesidad de ser programados de forma explícita para realizar dicha mejora

A continuación se retoma una definición de Aprendizaje Automático, que postula que se trata de un proceso de mejora al realizar una tarea vía la experiencia (Tom Mitchell) [35]:

Def. 3 *Aprendizaje Automático o de Máquina (AM) Se dice que un programa de computadora* aprende *de su experiencia (E) respecto a alguna clase de tareas (T) y una medida de rendimiento (MR), si su MR al realizar T mejora gracias a E.*

Ahora, se presenta brevemente el funcionamiento del AM y posteriormente se dan algunas consideraciones que influyen en este proceso de mejora automática del comportamiento.

Procesos involucrados en el AM

Como ya se dijo, el AM produce algoritmos, y con ellos se crean programas y herramientas que, posteriormente, se utilizan para diversas aplicaciones. El proceso de AA se da en dos fases:

1. Entrenamiento: se aplican los algoritmos a los datos, con el fin de obtener un *modelo*. Esto incluye una etapa de evaluación para conocer la bondad del algoritmo.
2. Aplicación: una vez que se obtuvo un modelo, éste es aplicado a la situación en cuestión, donde, de acuerdo con sus resultados, se alcanzará el objetivo de mejora de la ejecución de las actividades para las que se ideó.

En la Figura 3.5 se ilustran las dos etapas del AA.

Para llevar a cabo estas dos fases, comúnmente se utilizan bancos de datos o ejemplares, los cuales se dividen también en *Entrenamiento* y *Prueba*. Uno

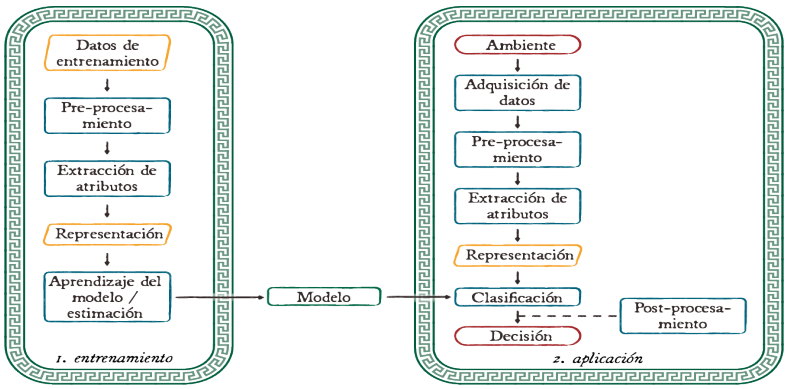


Figura 3.5: Las dos etapas del AM para obtener un modelo a partir de datos y algoritmos de aprendizaje.

de los objetivos del algoritmo es su capacidad de generalización, esto es, de obtener un modelo que funcione razonablemente con nuevos datos: si el algoritmo generaliza, podrá discriminar los datos que se le presentan que no pertenezcan a sus datos de entrenamiento.

De esta forma, los datos para entrenar y probar un algoritmo pueden presentárseles en al menos dos formas:

- División fija (conocida como *split*, en inglés): el banco de datos de tamaño $N(100\,\%)$ se divide en dos partes de acuerdo con porcentajes predefinidos. Esta forma es útil cuando está disponible una cantidad considerable de datos. Usualmente, este enfoque se emplea para comparar diferentes algoritmos de forma directa. Tiene dos subconjuntos de datos, a saber:

  - Entrenamiento e, que contiene el $n\,\%$ del universo de datos: se utiliza para generar el modelo con el algoritmo de aprendizaje en cuestión.
  - Prueba p, que contiene el m $\%$ del conjunto restante de datos: se utiliza para ajustar el modelo en construcción. Estos datos se presentan al modelo sólo al final, para medir su desempeño general. Es importante resaltar que que los datos de p **nunca sean vistos** por el modelo en la etapa de entrenamiento. Así se simulan datos que podrían venir del ambiente, como en un escenario real. Ver Figura 3.6.

- Rondas entrecruzadas de validación (conocida como *k-fold cross validation*, en inglés): en este método, se hacen k particiones de todo el conjunto de datos pero de forma alternada. De tal manera se realizan rondas e intercambios donde a cada partición le corresponde alternadamente hacer las veces de p, mientras el resto se comporta como e. Una vez concluido, se asigna una nueva partición y se vuelve a entrenar. En cada ronda siempre habrá un conjunto que no se use para entrenar el modelo. Así, se generan k modelos, y se escoge al mejor evaluado. Este método es útil cuando no se dispone de muchos datos para entrenar. Ver figura 3.7.

Figura 3.6: Método de partición fija para obtener los subconjuntos e y p a partir del banco de datos E. En este caso, $e = 33\,\%$ y $p = 66\,\%$.

Figura 3.7: Método de k rondas entrecruzadas de validación partir del banco de datos E, en esta caso: $k = 5$.

Elementos del AM

En suma, para conseguir implementar AM, se deben contemplar los siguientes elementos:

- La definición de lo que será aprendido (la clase de tarea), esto es, la definición de una **función de aprendizaje** que modele la experiencia (a través de alguna representación) y la utilice en la mejoría de la tarea a realizar (la **política de acción**).
- La definición de una **medida** o **función de rendimiento** que oriente el aprendizaje.
- El tipo de experiencia a tomar en cuenta.

Además, uno de los aspectos más importantes del AM tiene que ver con los **sesgos**, es decir, con aquello que lo influencia, pues decanta las posibilidades que éste tiene en cuanto a su capacidad de generar habilidades y mejorar las que ya posee. Lo sesgos en el AM se pueden categorizar como sigue:

- De representación: cómo están representados la experiencia y el conocimiento aprendido,... (sólo se puede aprender aquello que se puede representar).
- Del modelo: qué incluye y qué deja fuera el algoritmo (cuánto y cómo generaliza).
- Humanos o de contenido: qué información está sobre o infrarepresentada (cultura, género, ideología, clase social ...);implican o se derivan de la recolección, muestreo, anotación, disponibilidad, o apreciación humanos.

Por otro lado, el AM puede categorizarse de distinta manera, por ejemplo, por el tipo de retroalimentación que esté disponible. En este sentido, se puede hablar de tres tipos básicos de aprendizaje:

- Supervisado: aprende una función a partir de ejemplos de entradas y salidas (ya se sabe qué salida produce qué entrada, las respuestas están **etiquetadas**).
- No Supervisado: aprende a partir de **patrones** observados en la experiencia, pero no hay etiquetas que le indiquen éstos.
- Por refuerzo: es el más general, aprende a través de la interacción con el entorno y una evaluación del éxito de éstas, i.e. el **refuerzo** o **recompensa**.

Ahora bien, el AM se ha aplicado a algunas tareas específicas que extrapolan el ámbito de los agentes y los SMA. Dentro de éstas pueden contarse las siguientes, para las cuales hay distintos algoritmos:

- Clasificación: trata de inferir o estimar clases utilizando datos discretos o nominales.
  - Aplicaciones: detección de correo indeseado (*spam*), reconocimiento de imágenes, toma de decisiones, adjudicación de cateogorías, entre otros.
  - Algoritmos: regresión logística, árboles de decisión, clasificador bayesiano simple *Naïve Bayes*, discriminadores lineales, entre otros.
- Regresión: trata de inferir o estimar clases utilizando datos numéricos continuos.
  - Aplicaciones: precios de mercado, reconocimiento de imágenes y patrones.
  - Algoritmos: K vecinos más cercanos, kerneles gaussianos, máquinas de soporte vectorial, redes neuronales, entre otros.
- Agrupamiento: genera grupos o racimos con datos bajo ciertos criterios de similitud.
  - Aplicaciones: segmentación de mercados, agrupación de patrones, identificación de clases.
  - Algoritmos: K-medias (medianas o modas), agrupamiento jerárquico, propagación de afinidad, entre otros.

Además, buena parte de las aplicaciones se refieren al aprendizaje de conductas, para lo cual se emplean métodos basados en aprendizaje por refuerzo; entre algunos de estos algoritmos icónicos destaca *Q-Learning* [35]. Sin embargo, por brevedad no se abordarán en este volumen.

Planteamiento de una clasificación supervisada

Para ilustrar los temas anteriores sobre clasificación, a continuación se muestra un breve ejemplo del planteamiento de un problema de clasificación con aprendizaje supervisado en el dominio de la gestión de desastres, específicamente en el caso de sismos:

- Problema: cuando un nuevo sismo sucede, se genera una gran cantidad de información, por ejemplo, las noticias en los periódicos en línea, o lo que se publica en las redes sociodigitales, que es difícil de identificar, leer y analizar.
- Propuesta: este problema se puede solucionar intentando categorizar los textos de noticias relacionadas con sismos, con el objetivo de posteriormente realizar un análisis rápido de las distintas fuentes implicadas.
  - Datos requeridos: historial de textos del pasado etiquetados como sismo.
  - Objetivo: generar un modelo para clasificar notas sobre sismos aplicando las fases de entrenamiento, a partir del conjunto e, y prueba, para ajustar el modelo con los datos p.
  - Uso: utilizar el modelo aprendido para decidir si una nueva nota se refiere o no a tales eventos.

A continuación se caracterizan los agentes aprendices y los elementos que los integran.

4.3. Agentes que aprenden

Un agente capaz de incluir aprendizaje dentro de su operación, es decir, un Ag_{apr}, se compone de cuatro elementos generales [2]:

1. **Elemento Crítico**: evalúa la actuación del Ag y determina qué modificaciones hacer para mejorar la su actuación futura.
2. **Elemento de Aprendizaje**: responsable de realizar mejoras vía la experiencia y un algoritmo de AA. Se realimenta del Crítico para influir en los demás componentes.
3. **Elemento de Actuación**: responsable de seleccionar acciones externas; recibe estímulos y decide qué acciones emprender (es en sí un AgR).
4. **Elemento Generador de Problemas**: plantea acciones novedosas e informativas de los cuales el Ag_{apr} se beneficia (aunque no sean las más útiles en cierto momento). Son acciones exploratorias que lo llevan a descubrir sendas que podrían ubicarlo en mejor posición ante el problema a resolver.

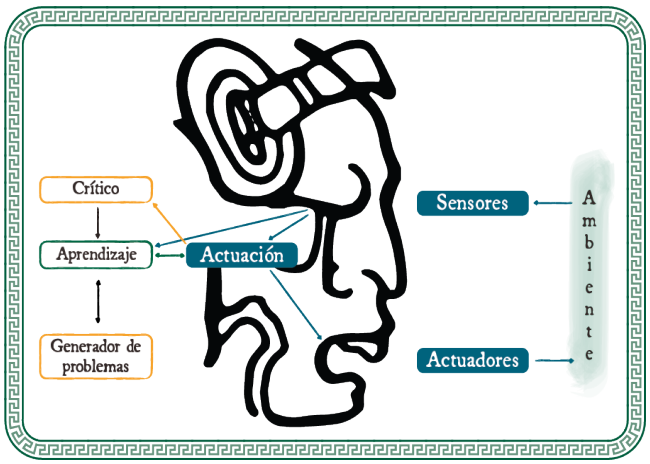


Figura 3.8: Agente que incorpora AM dentro de sus capacidades.

Ahora bien, ¿cómo programo una F_{Ag} para un Ag_{apr}? A continuación se puede ver el algoritmo base para implementar un Ag_{apr}.

```
Función agAprendiz(Percepción p): Acción
Var:
  ListaAcción A = {}, EstadoMundo em, Acción a,b, Estado ea, e
% Experiencia para realizar la tarea (inicialmente nula)
  Experiencia E
% Política de acción aprendida y evaluación del agente (inicializada ad hoc)
  Política l, Evaluación v
%%% Elem. Actuador 1a parte
  em <- determinarEdoMundo(p,ea)
  e <- actualizarEdoInterno(em)
% Busca la acción aplicable según e y la política de acción l
  si(l = {}) entonces:
    l <- aprende(e,v,E,l)
  a <- buscarAcciónAplicable(e,l,A)
%%% Elem. Generador de problemas: busca realizar acciones novedosas
  b <- probarNuevasAcciones(a)
  si(b != 0) entonces:
    a <- b
%%% Elem. Crítico
  v <- evaluaBeneficio(e,l,a,t)
%%% Elem. Aprendizaje: actualiza la experiencia y la política l
  E <- E U {p,a}
  l <- aprende(e,v,E,l)
%%% Elem. Actuador 2a parte
  responde(a)
```

5. Rendimiento y Aprendizaje

Una cuestión que va asociada conmúnmente con el AM tiene que ver con el rendimiento, tanto de los propios algoritmo de aprendizaje, como del agente que se vale de ellos para operar.

5.1. Recordando la Función de Rendimiento

Hay que revisar la FR en los agentes. Para definirla hay que expresar ciertas consideraciones sobre la MR.

- la MR Evalúa el *cómo*
- Su pregunta base es ¿qué tan exitoso ha sido un *Ag*? (tras una acción o serie de ellas).
- Debe ser objetiva.
- Es necesario diseñar medidas de rendimiento según lo que se quiere dentro del entorno, no con base en lo que se cree sobre cómo debe comportarse el *Ag*

Ahora bien, una vez que se ha diseñado una MR adecuada, es momento de ubicarla en el contexto en que está inserto el agente, esto es, en su entorno de trabajo.

5.2. Recordando los RAES

Esta es una forma de definir un entorno de trabajo para una agente. Diseñar un Ag_R pasa por especificar el **entorno de trabajo** lo más completo posible vía el RAES / REAS:

- **R**endimiento o Desempeño
- **A**mb o **E**ntorno
- **E**fectores o **A**ctuadores
- **S**ensores

A continuación se muestra un ejemplo de RAES. Dado el *AgR Tutor Inteligente de Música (TIM)*, situado en ambientes GNU-Linux/Unix, completaremos el siguiente RAES

Ag	R	A	E	S
`TIM`	Maximizar la puntuación de los alumnos en las evaluaciones	Sistema de archivos Unix y su sistema de tuberías (*pipeline*)	interfaz multimodal para desplegar contenidos, ejercicios, sugerencias, correcciones, evaluaciones	interfaz multimodal, teclado, ratón, otros

En cuanto a la Función de Rendimiento (FR) o **Función de Utilidad** (FU), se comenta lo siguiente:

- Asigna una cantidad numérica para expresar lo deseable de un estado: $FR : E \mapsto \Re$
- En combinación con la probabilidad de ocurrencia de las acciones, resulta la **utilidad esperada** UE.
- De UE puede calcularse omo una probabilidad condicional: $UE(a|e) = \sum_i P(Resultado_i(a)|Realizar(a), e) \bigcup (Resultado_i(a))$

Tomando como base, de la UE se deriva el principio de **máxima utilidad esperada**:

- La máxima utilidad esperada (MUE) se refiere a que:
  - Un Ag_R debiera elegir aquella acción que maximice su UE.
- Esto asemeja la FRcon la **Medida de Desempeño** por la cual se juzga su comportamiento:
  - Si la FR guía su comportamiento, tendrá el máximo desempeño posible $\Rightarrow$ la *racionalidad* como criterio global se reduce a un criterio local, específicamente a un elemento de diseño: la FR de un Ag_R.

¿Cómo sería la FR de estos Ag_R dadas las siguientes tareas y sus Medidas de Desempeño (MD)?

Ag_R	MD	**¿FR?**
Diagnóstico médico	Núm. de pacientes sanos (bien diagnosticados), costos, insumos utilizados, quejas	precisión y exactitud (Núm. de errores, falsos positivos/negativos), tiempo
Inspector de granos de café	kg/ton de granos bien evaluados, residuos identificados	precisión y exactitud
Tutor en línea	alumnos aprobados, % de cobertura de temarios	máxima puntuación de los pupilos, lecciones impartidas

5.3. Funciones de Aprendizaje

A continuación se muestra una breve descripción de cómo definir una Función de Aprendizaje. El AM parte de la definición de lo que será aprendido, esto es, la definición de una función de aprendizaje o Función Objetivo (FO).

Tarea	*FO*	**Experiencia**
Jugar damas inglesas	% de juegos ganados a los oponentes	Juegos de práctica contra sí mismo
Leer textos manuscritos	% de palabras clasificadas correctamente	Base de datos de textos manuscritos y su clasificación
Carro autónomo	promedio de viajes bien realizados (según reglamento de tránsito o juicios humanos	Base de datos multimodal de secuencias de manejo y las acciones tomadas (por un conductor humano experto)

Definición de un Problema de AM

El problema de AA se define determinando los elementos siguientes:

1. El tipo de experiencia E
2. La $FO : V$
3. La representación del conocimiento aprendido, es decir, la FO Aprendida (aproximada*), Hipótesis o Política de Acción: $\hat{V}$
4. El algoritmo de aprendizaje

*La función objetivo ideal V dista mucho de ser la que realmente se obtiene, por lo general ésta sólo se aproxima: $\hat{V}$

Ejemplo de la definición de una función objetivo

Se muestra a continuación cómo definir una FO para el caso de aprender a jugar damas inglesas [2]:

1. E: juegos frente a oponentes, juegos consigo mismo, tabla de movimientos correctos, ...
2. FO: movimientos en el tablero, valores en el tablero, ...
   - Sea la función $V : T \mapsto \Re$, donde los mejores registros corresponderían a configuraciones de tableros T más favorables
   - Así, la FO podría ser: $EscogeTirada : T \mapsto \Re$, que permite seleccionar un movimiento que conduzca a alguna de estas configuraciones favorables acorde con su evaluación numérica
3. La representación de la Función Aprendida, Hipótesis o Política de Acción:
   - Función polinómica acorde con ciertas características predefinidas del tablero o del juego
   - Red neuronal que reciba el vector indicado
   - Función lineal de 6 elementos $x_i, i = \{1, \ldots, 6\}$ (piezas negras y rojas, reyes rojos y negros, piezas negras amenazadas por rojas, y rojas amenazadas por negras) y sus respectivas ponderaciones de importancia, además de una para el tablero t en su conjunto $w_i, i = \{0, \ldots, 6\}$:

$$\hat{V}(t) = w_0 + w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + w_5x_5 + w_6x_6$$

4. El algoritmo de aprendizaje: descenso del gradiente, programación lineal, reglas o árboles de decisión.

   - Regla de actualización por mínimos cuadrados:

     `Por cada vector de entrenamiento` $\langle V_{entr}(t) \rangle$:

     - `Usar los pesos actuales para calcular` $\hat{V}(t)$
     - `Por cada peso` w_i, `actualizar como sigue`:
       $w_i \leftarrow w_i + \eta(V_{entr}(t) - \hat{V}(t))x_i$
       Donde η es una constante (digamos 0.1) para ajustar los pesos durante el aprendizaje

6. Ejemplos de agentes

6.1. Ejemplo 1: Triángulo, un agente tropista

Objetivo

Implementar un agente basado en las especificaciones de los Agentes Reflejos Simples para representar un ambiente tipo rejilla con un agente cuyo único propósito es navegar en él evadiendo obstáculos y alimentarse con frutillas que aparecen en su ambiente.

Desarrollo

Con base en las especificaciones de los Agentes Reflejos Simples:

- Basado en tabla o tropista: identifica ciertos estados del mundo con acciones, es decir realiza un mapeo ideal *percepción-acción*.

Se debe mplementar un agente "triángulo" con las características siguientes:

- Ambiente: una rejilla de *nxn* (determinada por el usuario), estático, parcialmente observable, episódico y discreto.

Debe contar además con:

- 10 % de casillas que contienen un obstáculo
- 15 % de casillas con comida

Ambas requieren estar dispuestas de manera aleatoria al inicio de la ejecución del agente.

También, cuenta con las habilidades siguientes:

- Se mueve sólo una casilla a la vez
- Se mueve un cuadro a la izquierda
- Se mueve un cuadro a la derecha
- Se mueve un cuadro abajo
- Se mueve un cuadro arriba

- Detectar comida
- Comer (hace desaparecer la comida del ambiente)
- Detectar obstáculos
- Evadir obstáculos

Para esta tarea, el funcionamiento del agente se basa en el objetivo de diseño del agente Triángulo, que es el siguiente:

- Buscar casillas con comida utilizando sus habilidades, cuando detecta un obstáculo, lo evade; si encuentra comida, la toma.

Considérese que aunque se acabe la comida, el agente continúa buscando, pues no lo sabe. La ejecución del agente puede terminar si el usuario presiona la tecla ESC.

Un ejemplo muy básico de este ambiente se puede ver en la Figura 3.9.

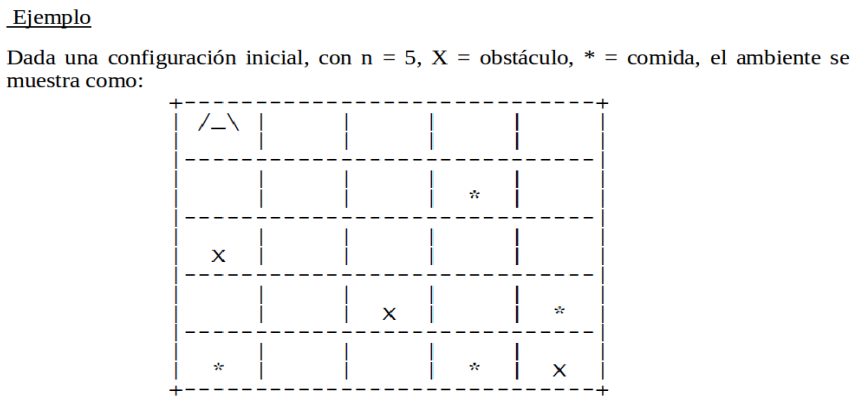


Figura 3.9: Representación básica del ambiente del agente Triángulo

Diseño de agente: RAES

Para definir al agente Triángulo como un agente tropista basado en una tabla, comenzaremos por conceptualizarlo mediante el Rendimiento, Ambiente, Efectores y Sensores, tal como se ve en la Tabla 3.1

Como se puede ver en el RAES, el ambiente es una rejilla donde el agente se sitúa de manera aleatoria, tal como la ubicación inicial de obstáculos y comida. Por otro lado, el agente Triángulo recorre el ambiente en actitud de forrajeo, por lo que consume la comida encontrada mientras evita obstáculos. Aunque el agente haya consumido toda la comida, seguirá explorando el ambiente. La

Ag	R	A	E	S
`Triángulo`	Núm. de obstáculos esquivados, Núm. de comidas tomadas/movimientos realizados	Tabla de nxn casillas: 15 % con comida, 10 % obstáculos, el resto estar vacías. La comida se regenera en algunas casillas ($<= 10\,\%$)	Función de forrajeo: permite moverse en `4-conectado` (arriba, abajo, derecha e izquierda) una casilla a la vez	Sensor de autoubicación en el ambiente; sensor de casillas: percibe 1 casilla alrededor en `4-conectado`; sensor de objetos en casilla (comida, obstáculos)

Tabla 3.1: Definición del agente Triángulo en términos de su RAES.

comida se regenera una vez consumida, apareciendo en casillas aleatoriamente, sin sobrepasar el límite porcentual. El usuario determina el tamaño del ambiente antes de que éste sea creado y de que se arranque el agente Triángulo.

Código y ejecución

El diseño del programa obedece a las funciones de agente definidas a lo largo de los temas de este capítulo, en concreto:

```
Función de agente reflejo simple o tropista, basado en tabla:

Función agRSBasadoEnTabla(Ambiente amb):
Var:
%Acción, inicializada con nulo/no acción
Acción a
%Aplica un algoritmo de mapeo percepción-acción
    Mientras(verdadero){
        percepcion <- percibir(amb)
        a <- selAccionAgRSTabla(p)
        amb = actuar(amb,a)
    }

Función selAccionAgRSTabla(Percepción p): Acción
Var:
%Acción, inicializada con nulo/no acción
Acción a
%Aplica un algoritmo de mapeo percepción-acción
    a <- buscaEnTabla(p)
    responde(a)
```

Además, la implementación que aquí se presenta tiene una función de demostración de los movimientos del agente para hacer más ilustrativo cómo se realiza ésta, por lo que incluye un menú que le pregunta al usuario si quiere desplegar la demo o ir directamente al agente basado en una tabla.

El programa de agente que implementa a Triángulo se puede ver en el listado 3.1 [3], codificado en el lenguaje de programación Python. Esta implementación utiliza la consola del sistema operativo para ejecutarse, no tiene interfaz gráfica sino que construye el ambiente como una rejilla en texto plano (código ASCII).

AgenteTriangulo1.py

```
# -*- coding: utf-8 -*-
"""
AgenteTriangulo1.py
Implementa el Agente Tri ngulo , un agente tropista basado en tabla
    .
Autor:
    Dr. Wulfrano Arturo Luna Ramirez
"""
import numpy as np
import time
import random
import os

# Utiler as del SO
# Windows
#clear = lambda: os.system('clear')
# GNU-Linux
clear = lambda: os.system('clear')
# Funci n de agente tropista
def agRSBasadoEnTabla(ambiente):
    # Implementa el ciclo: percibir-decidir-incidir
    movimientos = {
        'arriba': (-1, 0),
        'abajo': (1, 0),
        'izquierda': (0, -1),
        'derecha': (0, 1)
    }
    while True:
        time.sleep(1)
        despliegaAmb(ambiente,"Ag Reflejo Simple o Tropista o
    Basado en Tabla")
        # percibir: se ubica en el ambiente, obtiene p
        pos_actual = ubicaAg(ambiente)
        # decidir: buscar en la tabla qu  hacer para moverse
        #          a <- funci nDeMapeo(p), se mueve en el ambiente
        nueva_pos = selAccionAgRSTabla(movimientos,pos_actual)
        # incidir: se mueve en el ambiente (redibuja ambiente y
    agente)
        ambiente = nuevaPosicionAg(ambiente,pos_actual,nueva_pos)
# Funciones auxiliares
# Ambiente
def crearAmbiente(n):
    tablero = np.zeros((n, n), dtype=str)

    for i in range(n):
        for j in range(n):
            aleatorio = random.randint(1, 100)
            if aleatorio <= 10:
                tablero[i][j] = obstaculo  # Obst culo
            elif aleatorio <= 15:
                tablero[i][j] = '*'  # Comida
            else:
```

[3] Este programa, y una versión más gentil con el usuario se pueden encontrar en http://lab-ltsi.cua.uam.mx/Docencia/LibroIntroSMA-1/Codigos/

```
50                  tablero[i][j] = vacio  # Espacio vac o
51      return tablero
52  #
53  # Definici n de primitivas
54  #
55  # Ubica al agente dentro del ambiente
56  def ubicaAg(tablero):
57      # Encontrar la posici n inicial del agente
58      for i in range(len(tablero)):
59          for j in range(len(tablero[i])):
60              if tablero[i][j] == simAgente:
61                  pos_actual = (i, j)
62                  break
63      return pos_actual
64  #
65  # Habilidades
66  #
67  # Mover al agente
68  #  1. Se mueve s lo una casilla a la vez
69  #  2-5. Se mueve un cuadro a la izquierda, derecha, abajo y arriba
70  def mueveAg(movimientos,pos_actual):
71      # Selecciona aleatoriamente alguna direccion
72      direccion, (dx, dy) = random.choice(list(movimientos.items()))
73      #print(" Direcci n y coordanadas del pr ximo movimiento:",
        direccion, dx, dy)
74      print(" Dir y coords del pr x mov:",direccion, dx, dy)
75      # Calcula la nueva posici n
76      nueva_pos = (pos_actual[0] + dx, pos_actual[1] + dy)
77      # Regresa la nueva ubicaci n del agente
78      return nueva_pos
79
80  # Desplegar el ambiente que contiene al agente
81  def despliegaAmb(tablero,leyenda):
82      os.system("clear")
83      print("\n----------------------------------")
84      print("\n--- ",leyenda,"---")
85      print("\n----------------------------------\n")
86      print(tablero)
87  # Despliega al Agente en el ambiente
88  def despliegaAg(tablero,movimientos):
89      # Obtiene la posici n actual del agente
90      pos_actual = ubicaAg(tablero)
91      nueva_pos = mueveAg(movimientos,pos_actual)
92      return nuevaPosicionAg(tablero,pos_actual,nueva_pos)
93  # Calcula la nueva posici n del agente
94  def nuevaPosicionAg(tablero,pos_actual,nueva_pos):
95      # Cambia el tablero con la ubicaci n del agente dada
96      if 0 <= nueva_pos[0] < len(tablero) and 0 <= nueva_pos[1] < len
        (tablero[0]):
97          if tablero[nueva_pos[0]][nueva_pos[1]] != obstaculo:
98              tablero[pos_actual[0]][pos_actual[1]] = vacio
99              tablero[nueva_pos[0]][nueva_pos[1]] = simAgente
100             pos_actual = nueva_pos
101     return tablero
102
103 # Demo
104 # Ciclo de demostraci n de despliegue de un agente
105 def cicloDespliegaAmb(tablero,movimientos):
106     # Mover el agente aleatoriamente
107     while True:
108         time.sleep(1)
109         os.system("clear")
```

```
110         print("\n---------------------------------------")
111         print("\n------ Demo Ag reflejo movi ndose -----")
112         print("\n---------------------------------------\n")
113         print(tablero)
114         tablero = despliegaAg(tablero,movimientos)
115 #
116 # Funciones de Selecci n de Acci n
117 #
118 # Funci n de selecci n de acci n del agente tropista
119 # a <- funci nDeMapeo(p), regresa el mapeo acci n-percepci n
120 def selAccionAgRSTabla(movimientos,pos_actual):
121     # Editar aqu  si se quiere una selecci n de acci n distinta
122     accion = mueveAg(movimientos,pos_actual)
123     return accion
124 #
125 # DEMO
126 #
127 def agenteDemo(tablero):
128     movimientos = {
129         'arriba': (-1, 0),
130         'abajo': (1, 0),
131         'izquierda': (0, -1),
132         'derecha': (0, 1)
133     }
134     cicloDespliegaAmb(tablero,movimientos)
135 # Funci n general de agentes
136 def agente(opcion,ambiente):
137 # El usuario selecciona Demo
138     if opcion == 'D':
139         agenteDemo(ambiente)
140 # El usuario selecciona Agente Basado en Tabla
141     elif opcion == 'T':
142         agRSBasadoEnTabla(ambiente)
143     else:
144         print("Vuelva pronto!")
145 if __name__ == '__main__':
146     n = int(input("Ingrese el tama o del tablero: "))
147     # S mbolos:
148     simAgente = '\u25B2' #codigo de str para representar el agente
149     obstaculo = 'X' # Obst culo
150     comida = '*'  # Comida
151     vacio = ' '  # Espacio vac o
152     # Crea el ambiente
153     tablero = crearAmbiente(n)
154     tablero[0][0] = simAgente
155     print(tablero)
156     # Pide al usuario definir qu  opci n arrancar
157     print("Selecciona alguna opci n:\n")
158     print("Ver un demo de Agente movi ndose (escribe D)")
159     print("Agente simple basado en Tabla    (escribe T)")
160     opcion = input("> ")
161     # Arranca la opci n deseada
162     agente(opcion,tablero)
```

Código 3.1: Código de Triángulo

En las figuras 3.10 y 3.11 puede verse al agente Triángulo operando[4].

De acuerdo con la funcionalidad descrita, el agente Triángulo implementado en el programa 3.1 adolece de algunas omisiones. Se deja al lector como ejercicio

[4]Un video de su ejecución se puede encontrar en http://lab-ltsi.cua.uam.mx/Docencia/LibroIntroSMA-1/Codigos/

su identificación e implementación.

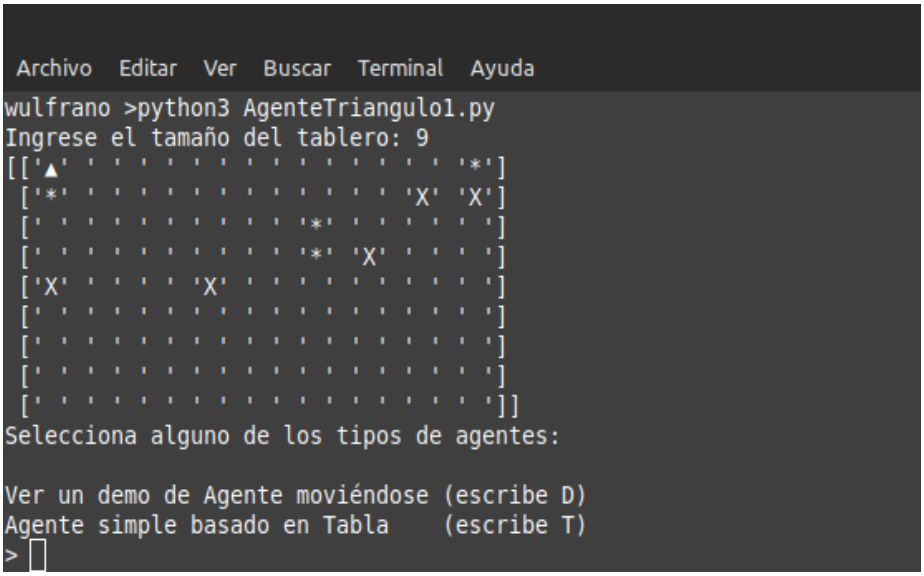


Figura 3.10: Agente Triángulo al iniciar: solicita al usuario el tamaño del ambiente y la funcionalidad a desplegar

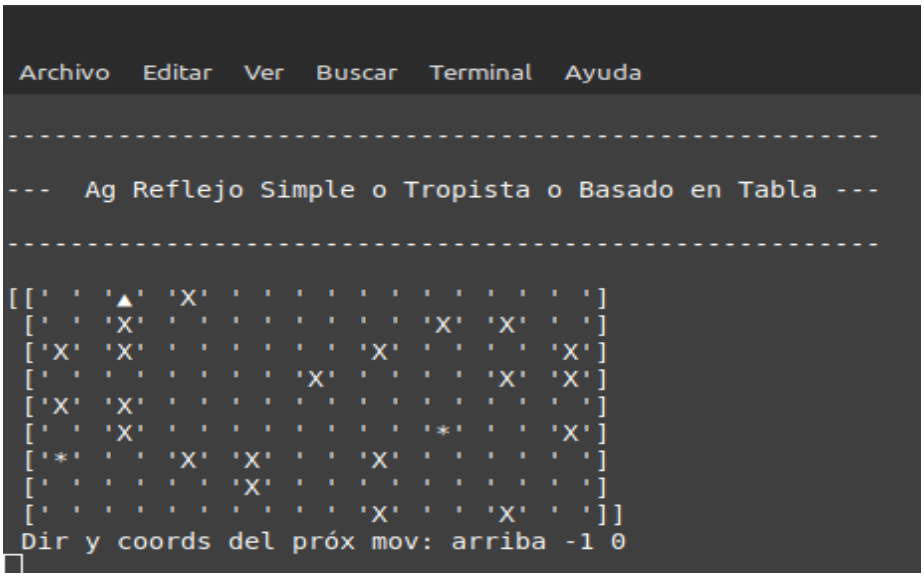


Figura 3.11: Agente Triángulo operando con su definición de selección de acción basada en una tabla

6.2. Ejemplo 2: el agente Leandro, un agente para armar

A continuación se presenta al *Agente Leandro*, un ejemplo de agente aprendiz que emplea el algoritmo *Naïve Bayes*.

Objetivo

Desarrollar una agente aprendiz que incorpore el algoritmo *Naïve Bayes* como parte de su mecanismo de decisión de acción para el conocido problema de jugar tenis.

El algoritmo *Naïve Bayes*

El *clasificador bayesiano simple o ingenuo*, mejor conocido como *Naïve Bayes*, es un algoritmo básico de Aprendizaje Automático, particularmente utilizado para realizar clasificación.

Este algoritmo construye un modelo generativo que trata de aproximar cómo se habrían producido los datos. Usa la *probabilidad a priori* de cada categoría (clase) sin pre-entrenamiento. El modelo de clasificación produce una distribución de *probabilidad a posteriori* de las posibles categorías dada la descripción del dato de entrada (el que se quiere clasificar).

En breve, a partir de un conjunto de datos, calcula las probabilidades de que un ejemplar pertenezca a una de las clases representadas en él. El algoritmo *Naïve Bayes*, de clasificación probabilística, se basa en el conocido Teorema de Bayes, expresado en la Fórmula 6.2.

$$P(C|X) = P(X|C)P(C)/P(X) \tag{3.1}$$

Donde:

$P(C)$: probabilidad de ocurrencia de ejemplares pertenecientes a la clase C

$P(X)$: probabilidad de ocurrencia del ejemplar observado X

$P(X|C)$: probabilidad observada de que el ejemplar X esté etiquetado con la clase C

$P(C|X)$: probabilidad condicional de que un ejemplar observado X pertenezca a la clase C

El algoritmo parte de la suposición de independencia entre las variables para calcular la probabilidad de ocurrencia de cada clase C.

De esta forma, con este algoritmo de Aprendizaje Automático y datos de ejemplo sobre una tarea se crea un modelo (es decir, las estadísticas que se obtienen una vez que haya sido entrenado con el conjunto de entrenamiento) que puede ser aprovechado por un agente aprendiz, tal es el caso del agente Leandro.

Desarrollo

Aquí se plantea una versión mínima del agente Leandro, enfocada en mostrar cómo se puede utilizar un algoritmo de Aprendizaje Automático para resolver una tarea: indicar si ciertas características del día son propicias para jugar tenis.

De esta forma, el agente inquiere al usuario los datos del día actual para saber si es adecuado salir a jugar tenis (o cualquier otro deporte al aire libre). Los indicadores de la bondad climática son agrupados en preguntas sobre diversas condiciones climáticas, cuyos valores se indican con un número entero para facilitar su procesamiento con *Naïve Bayes*:

- Pronóstico Climático: Soleado (1), Despejado (2), Lluvioso (3)
- Temperatura: Cálido (1), Templado (2), Frío (3)
- Humedad: Alta (1), Normal (2)
- Viento: Fuerte (1), Débil (2)

De este modo, para un día caraterizado con condiciones como las siguientes:

- Pronóstico Climático: Soleado (1)
- Temperatura: Cálido (1)
- Humedad: Alta (1)
- Viento: Fuerte (1)

se recomendaría no salir a jugar.

En cuanto al diseño del agente, el entorno de trabajo resulta bastante sencillo, pues el ambiente es la consola del sistema operativo y su interacción con el usuario se basa en preguntas y respuestas. Por ello, se deja al lector la definición del RAES para este agente; sólo se pide tomar en consideración que el mismo incluye el elemento de Aprendizaje Automático como una habilidad del agente.

Leandro fue implementado en Python, y para emplear Aprendizaje Automático se utilizó el paquete de análisis predictivo de datos ScikitLearn [36] escrito también en Python.

El código del agente Leandro se muestra en el listado 3.2 [5].

`agLeandro-V0.py`

```
# -*- coding: utf-8 -*-
'''
agLeandro-V0.py
Implementa un agente aprendiz (usando el algoritmo Naive Bayes) con
     ScikitLearn
 Autor:
 Dr Wulfrano Arturo Luna Ram rez
'''
import sys
import time
import os
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
```

[5] El código y un video de su ejecución se puede encontrar en LIGA-A-VIDEOS PENDIENTE

```
14
15 # Utiler as del SO
16 # Windows
17 #clear = lambda: os.system('clear')
18 # GNU-Linux
19 clear = lambda: os.system('clear')
20 #
21 # Funciones auxiliares de Aprendizaje Autom tico
22 #
23 #
24 # Datos tennisNumericoClase.csv
25 #
26 # Formato:
27 #        E, E, E, E, E, C
28 # E = {1,2,3}
29 # C = {1,0}
30 #
31 # Carga el archivo de entrenamiento de Tenis
32 def cargaDatos(archivo):
33 #    nomArch = '../datos/tennisNumerico.csv'
34     f=open(archivo)
35     data = np.loadtxt(f,delimiter=',')
36     clase = data[:,4:]
37     clase = np.squeeze(clase)
38     print(clase)
39     # Realiza las particiones para generar el modelo:
40     # datosE = conjunto de entrenamiento
41     # datosP = conjunto de prueba
42     # claseE = conjunto de clases de datosE
43     # claseP = conjunto de clases de datosP
44     # Crea los conjuntos de prueba y entrenamiento antedichos
45     datosE, datosP, claseE, claseP = train_test_split(data, clase,
       test_size=0.5, random_state=0)
46     print(" datosE = conjunto de entrenamiento \t [%d] renglones"
       %(len(datosE)))
47     print(" datosP = conjunto de prueba \t\t [%d] renglones " %(len
       (datosP)))
48     print(" claseE = conjunto de clases de claseE \t [%d] renglones
        " %(len(claseE)))
49     print(" claseP = conjunto de clases de claseP \t [%d] renglones
        " %(len(claseP)))
50     print(" \t\t\t Total =\t [%d] renglones " %(len(datosE)+len(
       datosP)))
51     return datosE, datosP, claseE, claseP
52 # Entrenamiento del algoritmo indicado (Naive Bayes) con los datos
       provistos
53 def entrenarNB(algAA,datosE, datosP, claseE, claseP):
54     # Crea el modelo a partir de los datos y clases de
       entrenamiento
55     modelo = algAA.fit(datosE, claseE)
56     # Realiza predicciones con los datos
57     pred = modelo.predict(datosP)
58     numReng = datosP.shape[0]
59     numBienEtq = (claseP == pred).sum()
60     numMalEtq = (claseP != pred).sum()
61     print("No. de casos de prueba : %d" % (numReng))
62     print("No. de casos BIEN etiquetados:  %d / %s" % (numBienEtq,
       numReng))
63     print("No. de casos  MAL etiquetados:  %d / %s" % (numMalEtq,
       numReng ))
64     return modelo
65 # Predicci n de un ejemplar a partir de un modelo entrenado
```

```
66 def predecirNB(ejemplar,modelo):
67     return modelo.predict(ejemplar)
68 # Solicita datos al usuario
69 def capturaTenisUsuario():
70     print(" :) #  Introduce los datos del d a de hoy para saber si
        vamos a jugar tenis ")
71     print(" :) #  Usa el siguiente formato <<obligatorio>>")
72     print(" :) #    Pron stico Clim tico: Soleado 1, Despejado 2,
        Lluvioso 3")
73     print(" :) #    Temperatura: C lido 1, Templado 2, Fr o 3")
74     print(" :) #    Humedad: Alta 1, Normal 2")
75     print(" :) #    Viento: Fuerte 1, D bil 2")
76     print(" :) #  Ejemplo 1: para un d a caraterizado como:")
77     print(" :) #    Pron stico Clim tico: Soleado")
78     print(" :) #    Temperatura: C lido")
79     print(" :) #    Humedad: Alta")
80     print(" :) #    Viento: Fuerte")
81     print(" :) #  Se considera una entrada de datos como sigue:")
82     print(" :) #  Para representar los valores nominales:Soleado,
       Caliente, Alta, Fuerte = D a no apto para jugar tenis")
83     print(" :) #  Se deben ingresar los valores num ricos: 1, 1,
       1, 1")
84     print(" :) #")
85     print(" :) #  Ahora dame tus datos respondiendo a cada pregunta
       ")
86     pronostico  = input("Pron stico clim tico?: ")
87     temperatura = input("Temperatura?: ")
88     humedad     = input("Humedad: ")
89     viento      = input("Viento?: ")
90     return pronostico,temperatura,humedad,viento
91 def rotuloBievenida():
92         print("################################################")
93         print("#                                              #")
94         print("# Hola soy el agente aprendiz LEANDRO  :-)     #")
95         print("#  Se decidir cu ndo ir a jugar tenis.         #")
96         print("# Mi funci n de decisi n usa el algoritmo      #")
97         print("# Naive Bayes, un m todo sencillo pero efectivo#")
98         print("# para la clasificaci n y la toma de decisiones#")
99         print("################################################")
100        print("# Un algoritmo de clasificaci n probabil stica#")
101        print("# que se basa en el Teorema de Bayes:          #")
102        print("#               P(C|X) = P(X|C)P(C)/P(X)       #")
103        print("# supone independencia entre las variables para #")
104        print("# calcular la probabilidad de cada clase.      #")
105        print("#                                              #")
106        print("################################################")
107 #
108 # FUNCIONES DE AGENTE
109 #
110 # Funci n simple de agente aprendiz
111 # Implementa el uso de un algoritmo de aprendizaje para resolver
       una tarea
112 def agAprendiz():
113     # Implementa el ciclo: percibir-decidir-incidir
114     modelo = ""
115     while True:
116         time.sleep(1)
117         os.system("clear")
118         rotuloBievenida()
119         # percibir: se ubica en el ambiente, obtiene p
120         percepcion = percibir()
121         # decide e incide
```

```
122         modelo = decidirIncidir(percepcion,modelo)
123         print("Pulsa cualquier tecla para continuar...")
124         opcion = input("> ")
125 # Funci n de percepci n
126 def percibir():
127     print("Estoy aqu  para atenderte,  qu  necesitas?:\n")
128     print("Generar un nuevo modelo a partir de datos (escribe N)")
129     print("Predecir un nuevo caso para jugar tenis  (escribe J)")
130     print("Terminar mi ejecuci n                    (escribe T)")
131     opcion = input("> ")
132     return opcion
133 # Decide lo que el agente debe hacer en cada iteraci n del ciclo
    percibir-decidir-incidir
134 def decidirIncidir(perc,modelo):
135     if perc == 'N':
136         print(" :) #  Entrenando ")
137         # Define el algoritmo a usar de la biblioteca SciKitLearn:
    Naive Bayes
138         algoritmoAA = GaussianNB()
139         # Carga los datos de entrenamiento
140         nomArch = '../datos/tennisNumericoClase.csv'
141         print(" :) #    cargando archivo ",nomArch," ...")
142         datosE, datosP, claseE, claseP = cargaDatos(nomArch)
143         # Obtiene el modelo a partir de los datos y el algoritmo
    indicado
144         print(" :) #    obteniendo modelo...")
145         modelo = entrenarNB(algoritmoAA, datosE, datosP, claseE,
    claseP)
146         # Incidir en el ambiente
147         print(" :) #    ya se ha obtenido el modelo")
148     elif perc == 'J':
149         if modelo == "":
150             print("Para usar un modelo, primero debes entrenarlo")
151         else:
152             p,t,h,v = capturaTenisUsuario()
153             ejemplar = [int(p),int(t),int(h),int(v),0]
154             print(" :) #  Haciendo predicci n")
155             # Aplica el modelo obtenido en el entrenamiento al
    ejemplar nuevo
156             prediccion = predecirNB([ejemplar],modelo)
157             # Incidir en el ambiente/usuario
158             if prediccion[0] == 1:
159                 print(" :) # ( + ) S  se puede ir a jugar tenis,
    prepara tus raquetas")
160             else:
161                 print(" :( # ( - ) NO se puede ir a jugar tenis,
    espera tiempos mejores")
162     elif perc == "T":
163         print(" :) # Hasta pronto ")
164         exit(0)
165     else:
166         print(" :S # Opci n no reconocida, intenta de nuevo ")
167 
168     return modelo
169 #
170 # Funcion principal
171 #
172 def main():
173     try:
174         agAprendiz()
175     except Exception:
176         traceback.print_exc(file=sys.stdout)
```

```
177     sys.exit(0)
178
179 if __name__ == '__main__':
180     main()
```

Código 3.2: Código del Agente Leandro.

En las figuras 3.12, 3.13 y 3.14, se puede ver a Leandro operando en su ciclo: *percibir-decidir-incidir*.

En la figura 3.13 se observa al agente Leandro en una interacción mínima con el usuario: cuando el usuario ya hizo una selección de las opciones que le presenta, la generación de un modelo de AM basado en su único algoritmo disponible, *Naïve Bayes*.

En la figura 3.14 se ve la culminación de la operación del agente, tras el uso de su modelo de AM y la presentación de la respuesta solicitada por el usuario.

```
########################################################################
#                                                                      #
# Hola soy el agente aprendiz LEANDRO  :-)                             #
#  Se decidir cuándo ir a jugar tenis.                                 #
# Mi función de decisión fue desarrollada con el algoritmo             #
# Naive Bayes, un método sencillo pero efectivo                        #
# para la clasificación y la toma de decisiones.                       #
########################################################################
# Naive Bayes es un algoritmo de clasificación probabilística.         #
# Se basa en el Teorema de Bayes:                                      #
#               P(C|X) = P(X|C)P(C)/P(X)                               #
# en la suposición de independencia entre las variables para           #
# calcular la probabilidad de cada clase.                              #
#                                                                      #
########################################################################
Estoy aquí para atenderte, ¿qué necesitas?:

Generar un nuevo modelo a partir de datos (escribe N)
Predecir un nuevo caso para jugar tenis   (escribe J)
Terminar mi ejecución                     (escribe T)
>
```

Figura 3.12: Agente Leandro al iniciar: se autopresenta y solicita la funcionalidad a desplegar.

```
########################################################################
#                                                                      #
# Hola soy el agente aprendiz LEANDRO  :-)                             #
#  Se decidir cuándo ir a jugar tenis.                                 #
# Mi función de decisión fue desarrollada con el algoritmo             #
# Naive Bayes, un método sencillo pero efectivo                        #
# para la clasificación y la toma de decisiones.                       #
########################################################################
# Naive Bayes es un algoritmo de clasificación probabilística.         #
# Se basa en el Teorema de Bayes:                                      #
#               P(C|X) = P(X|C)P(C)/P(X)                               #
# en la suposición de independencia entre las variables para           #
# calcular la probabilidad de cada clase.                              #
#                                                                      #
########################################################################
Estoy aquí para atenderte, ¿qué necesitas?:

Generar un nuevo modelo a partir de datos (escribe N)
Predecir un nuevo caso para jugar tenis   (escribe J)
Terminar mi ejecución                     (escribe T)
> N
 :) #  Entrenando
 :) #    cargando archivo  ../datos/tennisNumericoClase.csv  ...
[0. 0. 1. 1. 1. 0. 1. 0. 1. 1. 1. 1. 1. 0.]
 datosE = conjunto de entrenamiento      [7] renglones
 datosP = conjunto de prueba             [7] renglones
 claseE = conjunto de clases de claseE   [7] renglones
 claseP = conjunto de clases de claseP   [7] renglones
                        Total =          [14] renglones
 :) #    obteniendo modelo...
No. de casos de prueba : 7
No. de casos BIEN etiquetados:  7 / 7
No. de casos  MAL etiquetados:  0 / 7
 :) #    ya se ha obtenido el modelo
Pulsa cualquier tecla para continuar...
>
```

Figura 3.13: Agente Leandro operando una vez que el usuario ha seleccionado *N*, es decir generar un nuevo modelo de *Naïve Bayes*.

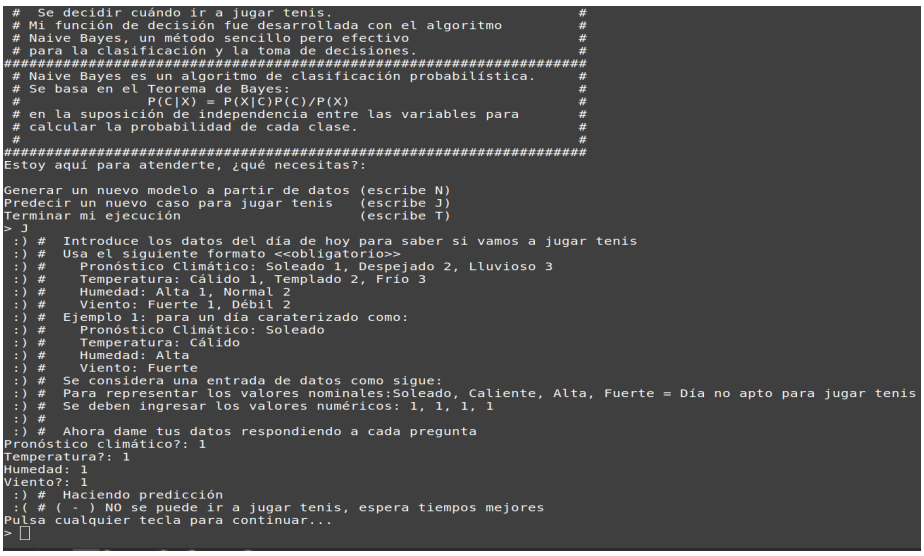


Figura 3.14: Agente Leandro operando una vez que el usuario le pide un pronóstico para salir a jugar con un caso negativo

Como puede apreciarse, este agente es una versión minimalista, al igual que el agente Triángulo. De este modo, Leandro presenta un interacción muy modesta, no gráfica, sino basada en una consola. Sin embargo, no debe distraerse el lector con su interfaz tan primitiva, lo que importa es que este agente utiliza un modelo de aprendizaje para contestar una tarea, que aunque sencilla, representa un reto no tan trivial si no se aplicara un método como este en su operación.

7. Resumen

Agentes racionales

En este capítulo se describieron brevemente los agentes racionales bajo una breve clasificación que atiende a la conformación de su programa de agente, siguiedo la premisa: $Agente = Arquitectura + Programa$

- Hay dos nociones básicas de agente artificial: *fuerte* y *débil*.
- La racionalidad en agentes se puede caracterizar mediante cuatro fatores: acciones, conocimiento, secuencia perceptual y función de rendimiento.
- Entre los rasgos más relevantes de los agentes racionales están la autonomía, su interacción y capacidad de aprendizaje.
- El ambiente, medio físico o informático donde se encuentran los agentes tiene varios tipos y características, donde los parcialmente observables, no episódicos y dinámicos son de los más retadores para el desarrollo de agentes (tal como el mundo real).
- El concepto de RAES es una descripción útil para iniciar el diseño de agentes, pues pone de manifiesto tanto el rendimiento del agente como los elementos de interacción entre éste y el ambiente.
- El Aprendizaje Automático es producto de ciertos algoritmos, datos y procesos que se dividen en entrenamiento y aplicación. En su aplicación a los agentes, se ve intrínsecamente vinculado con el concepto de rendimiento.
- Durante el capítulo, se dieron las bases para implementar distintos programas de agente: reactivos (reflejo simple, basdo en modelos, en objetivos y en utilidad) y cognitivos, además de los agentes que incorporan Aprendizaje Automático.

8. Actividades de aprendizaje

8.1. Reforzamiento cognitivo

Elabore las siguientes tareas para reforzar lo aprendido.

- Realice un mapa conceptual con los conceptos clave del capítulo.
- Lleve a cabo una búsqueda documental sobre sistemas desarrollados como agentes en los diferentes sectores: educativo, industrial, salud, servicios, entre otros.
- Tomando en cuenta estas características, se deja al lector probar a clasificar los siguientes sistemas como programas o agentes:

  1. Un reloj
  2. Un termostato
  3. Un teléfono celular
  4. Control de acceso para personas
  5. Aire acondicionado de un automóvil
  6. Una lavadora
  7. Un tutorial digital para enseñar música
  8. La vigilancia en una librería para evitar que alguien saque un producto sin pagar
  9. Internet
  10. La red telefónica

8.2. Ejercicios de programación

Agente Triángulo

- Objetivo: modificar al agente tropista Triángulo definiendo su función de agente como:
  - Basada en función.
  - Basada en reglas.
- Desarrollo: se deben realizar las siguientes tareas:
  - Análisis del Entorno de Trabajo (RAES).
  - Implementar las funciones de agente solicitadas y añadirlas al código del agente Triángulo.
  - Implementar un menú de arranque del agente, para que el usuario determine, además del tamaño del ambiente, qué función de agente debe emplear Triángulo para funcionar.
  - Verificación de su funcionamiento.

Agente Leandro

- Objetivo: modificar al agente aprendiz Leandro, para mejorar sus habilidades de interacción con el usuario y trasladarlo a un dominio distinto.
- Desarrollo: se plantean dos etapas; cada una introduce un elemento de aprendizaje y agrega características al agente a fin de hacerlo más operativo.
  - Etapa 1: modificar el programa `agLeandro-V0.py` (renombrarlo como `agLeandro-V1.py`) para que genere el modelo entrenado y lo guarde para ser usado posteriormente sin tener que entrenarse cada vez que se arranca (salvo que se añadan más datos, es decir, un nuevo archivo CSV). El modelo se puede guardar en un archivo o en una estructura de datos que Leandro ocupe mientras opera.
  - Etapa 2: modificar el programa que usa Leandro (los programas anteriores) para que se traslade al dominio del agente Triángulo (`agLeandro-V2.py`):
    - Análisis del Entorno de Trabajo (RAES).
    - Generar una tabla de datos que represente el ambiente del agente Triángulo con base en 4 u 8 conectado. Cada renglón indicaría qué debe hacer el agente en cada caso, y si en él hay comida u obstáculos. Esto es, generar el conjunto de entrenamiento para obtener el mecanismo de decisión basado en *Naïve Bayes*.
    - Incorporar el modelo resultante de entrenar *Naïve Bayes* al agente Leandro para que pueda operar en el ambiente del agente Triángulo.

8.3. Cuestionario 3

Responda las siguientes preguntas sobre Agentes Racionales[6].

1. La reactividad en agentes se refiere a:

   a) Su capacidad de respuesta a cambios en el entorno

   b) El uso de reglas simples

   c) La búsqueda de objetivos

   d) La continuidad en su comportamiento

   e) La interacción con otros agentes

2. ¿Qué implica la reactividad respecto al ambiente?

   a) Ignorar las condiciones cambiantes del entorno

   b) Tener en cuenta el posible fracaso en entornos dinámicos

   c) No requerir actualización ante cambios

   d) Funcionar siempre en lazo abierto

   e) Reaccionar solo a entornos estáticos

[6]Este cuestionario fue realizado parcialmente con una IA Generativa. Ver el esquema de trabajo del apéndice C.

3. Los sistemas reactivos ...

   a) No responden a cambios en el entorno
   b) Requieren representación explícita del mundo
   c) Interactúan constantemente con el entorno
   d) Planifican secuencias de acciones complejas
   e) Se basan únicamente en conocimiento previo

4. La proactividad en agentes se refiere a ...

   a) Su capacidad de iniciativa y logro de metas
   b) Su reacción a estímulos del entorno
   c) Su funcionamiento continuo en el tiempo
   d) Su habilidad para interactuar con otros
   e) Su aprendizaje a partir de experiencias

5. Un agente proactivo ...

   a) Es completamente reactivo
   b) Actúa sólo cuando se le indica
   c) Reconoce y aprovecha oportunidades
   d) Carece de objetivos propios
   e) Requiere supervisión externa

6. La proactividad implica que el agente ...

   a) Reacciona a cambios en el entorno
   b) Interactúa con otros agentes
   c) Aprende nuevas habilidades
   d) Tiene un comportamiento persistente
   e) Toma iniciativas necesarias

7. La autonomía en agentes artificiales se refiere a ...

   a) Su capacidad de establecer metas propias
   b) Que su comportamiento depende de su experiencia
   c) Su habilidad para interactuar con humanos
   d) Que funcionan sin supervisión externa
   e) Que aprenden de ejemplos no estructurados

8. ¿Qué implica mayor autonomía en un agente?

   a) Comportamiento basado en reglas fijas
   b) Reacción directa a percepciones
   c) Dependencia de conocimiento previo
   d) Aprendizaje a partir de experiencias

e) Funcionamiento aislado de otros

9. Un agente autónomo ...

   a) Requiere reprogramación frecuente
   b) Depende de supervisión externa
   c) Basa su comportamiento en aprendizaje
   d) No tiene iniciativa ni metas propias
   e) Es controlado remotamente

10. La sociabilidad en un agente se refiere a ...

    a) Su habilidad para interactuar con otros agentes
    b) Su funcionamiento continuo en el tiempo
    c) Su capacidad de establecer metas propias
    d) Su aprendizaje a partir de experiencias
    e) Su reacción a cambios en el entorno

11. La sociabilidad implica que el agente ...

    a) Trabaja de forma aislada
    b) Busca cooperar con otros agentes
    c) No requiere comunicarse
    d) Reacciona solo a percepciones
    e) Carece de habilidades sociales

12. Un agente sociable ...

    a) Rechaza la interacción con otros
    b) Usa protocolos de comunicación
    c) Tiene comportamiento no persistente
    d) Le falta iniciativa propia
    e) No establece relaciones de cooperación

13. La persistencia en agentes inteligentes se refiere a ...

    a) Su aprendizaje constante
    b) Su continuidad comportamental
    c) Sus habilidades sociales
    d) Su capacidad de establecer metas
    e) Su reacción a percepciones

14. Un agente persistente ...

    a) Tiene un comportamiento discontinuo
    b) No sigue un ciclo percibir-decidir-actuar
    c) Carece de objetivos propios

d) Se rinde fácilmente ante dificultades

e) Mantiene su funcionamiento orientado a metas

15. La persistencia implica que el agente ...

a) Abandona sus objetivos con facilidad

b) Interactúa con otros agentes

c) Cambia de tareas constantemente

d) Reacciona a estímulos del entorno

e) Continúa tratando de lograr sus metas

16. El aprendizaje en un agente inteligente se refiere a ...

a) Su sociabilidad con otros agentes

b) Su continuidad en el comportamiento

c) Su habilidad para adquirir nuevo conocimiento a partir de la experiencia

d) Su capacidad de reaccionar al entorno

e) Sus respuestas basadas en reglas fijas

17. Un agente que aprende ...

a) Se basa solo en conocimiento preprogramado

b) No cambia su comportamiento con la experiencia

c) Tiene un funcionamiento discontinuo

d) Mejora su rendimiento con el tiempo

e) No generaliza comportamientos

18. El aprendizaje permite a un agente ...

a) Ser menos autónomo

b) Reforzar su racionalidad limitada

c) Reducir su capacidad de adaptación

d) Ignorar nueva información

e) Adquirir nuevo conocimiento útil

19. La racionalidad en un agente inteligente se refiere a ...

a) Su capacidad de aprendizaje constante

b) Su habilidad para establecer metas propias

c) Su reacción a cambios en el entorno

d) Que realiza acciones correctas para sus objetivos

e) Su continuidad en el comportamiento orientado al logro de sus metas

20. Un agente racional ...

a) Escoge acciones al azar

b) Cambia de tareas frecuentemente

c) Abandona fácilmente sus metas

d) Hace lo correcto para tener éxito

e) No considera los resultados de sus acciones

21. La racionalidad implica que el agente ...

a) Maximiza una medida de rendimiento

b) No aprende de su experiencia

c) Carece de objetivos propios

d) Reacciona sólo ante percepciones

e) No requiere representación del entorno

22. El ambiente de un agente se refiere a ...

a) El espacio físico o virtual donde está situado

b) Otros agentes con los que interactúa

c) Sus objetivos internos

d) Su continuidad comportamental

e) Sus capacidades de aprendizaje

23. En relación al ambiente, un agente racional ...

a) Lo ignora y actúa de forma aislada

b) Debe procurar ambientes simulados

c) No requiere representarlo

d) Es la solución para los problemas del ambiente

e) No necesita adaptarse a cambios

24. El ambiente de un agente debe permitir ...

a) Supervisión externa completa

b) Comportamiento discontinuo

c) Ausencia de autonomía

d) Persistencia y autonomía del agente

e) Acciones independientes de percepciones

25. Desde el ambiente, un agente realiza ...

a) Inferencias para ignorar el ambiente

b) Percepción, decisión y acción continuas

c) Reprogramación periódica

d) Solo reacción a estímulos

e) Acciones sin interpretar percepciones

26. Para un agente que interactúa con su ambiente ...

    a) La percepción no es relevante
    b) No existen problemas a resolver
    c) No requiere tomar acciones
    d) El ambiente se mantiene siempre estático
    e) Debe interpretar lo que percibe

27. Un agente que actúa en su ambiente ...

    a) No afecta las condiciones del ambiente
    b) Se abstrae completamente del ambiente
    c) Sólo reacciona a los cambios
    d) Realiza acciones independientes de sus metas
    e) Busca modificar el ambiente hacia sus metas

Capítulo 4

Búsqueda en IA

Objetivos

- Conocer la formulación de problemas
- Conocer los métodos de búsqueda en espacio de estados, informados y no informados
- Programar búsquedas informadas y no informadas utilizables en agentes

¿Por qué se equivocaron los científicos? «Por los prejuicios, sobre todo —dijo Allie, enroscada alrededor de Gibreel debajo de la seda del paracaídas—. Puesto que no pueden cuantificar la voluntad, la dejan fuera de sus cálculos»

Salman Rushdie, *Los versos satánicos*

A un agente se le plantea constantemente el dilema de ¿cómo alcanzar una meta específica?, lo que configura un escenario de *espacio de búsqueda* de la solución a esa meta. Este agente nace como respuesta ante la necesidad de alcanzar una meta a través de una secuencia de acciones. Como se plantea en el capítulo 3, el mapeo del agente es el elemento crucial para definir su comportamiento (y es el cometido de su construcción), además de que realizar este mapeo no es una tarea trivial. Por ello, en un intento por simplificar el diseño, se plantea el logro de metas como objetivo del agente. Nace así la propuesta del Agente Basado en Objetivos ya introducida en el capítulo antedicho.

En este capítulo se revisarán más a fondo las características de estos agentes y se proporcionarán los elementos mínimos para programarlos.

1. Búsqueda en espacio de estados

Para complementar el diseño de un Agente Basado en Objetivos (Ag_{obj}) es necesario analizar el tema de búsqueda en un espacio de estados. Entonces, el diseño de un Ag_{obj} que resuelve un problema, pasa por determinar las siguientes cuestiones:

- ¿Cómo se formula un problema?
- ¿Cómo se resuelve éste una vez formulado?
- ¿Cómo se implementa un criterio de búsqueda de una solución o la consecución de metas?

Diseño de un Ag_{obj}: formulación de un problema

- El diseño de un Ag_{obj} se puede ver como la repetición de las siguientes actividades:
  - Formular Metas, Formular Problemas para lograr cada Meta, Buscar la solución, Ejecutarla
- De manera sucinta:
  - Problema = {meta m, {M medios que permiten alcanzarla}}
  - Búsqueda = {procedimiento de exploración aplicable a M}

Problemas y su formulación

- De acuerdo con las definiciones (3 y 5) que nos ofrece la *Real Academia Española* [1], un problema es:

[1] https://dle.rae.es/problema

3. Conjunto de hechos o circunstancias que dificultan la consecución de algún fin.
5. Planteamiento de una situación cuya respuesta desconocida debe obtenerse a través de métodos científicos.

- En ambos casos, nos remite a una dificultad (o un conjunto de ellas) que hay que superar al perseguir un objetivo determinado -el fin o la respuesta buscada.

Es así que, para superar esta dificultad se deben tomar acciones y allegarse los medios necesarios para responder adecuadamente. A continuación se hace una pequeña mención entre los tipos de problemas, de acuerdo con lo expuesto en [8]:

- *Estructura inducida*: se conocen los elementos del dominio y las partes necesarias para resolver el problema, la solución es un patrón o regla de articulación sistemática de toda la estructura.
- *De combinación*: se resuelven mediante el arreglo de las partes constitutivas, es decir, hallar una combinación u orden pertinente (que es la solución).
- *De transformación*[2]: se conocen los estados *inicial* y *final* (la meta, la solución), y las operaciones necesarias para pasar de uno al otro. La dificultad consiste en aplicar las operaciones en un orden adecuado.

A continuación, se mencionan dos clases de problemas, de particular relevancia por sus diferencias, ya que han dado lugar a diferentes enfoques en la IA.

Problemas reales y modelos

Los problemas pueden dividirse en dos categorías: los del mundo real y los problemas de juguete. En el primer caso, se menciona que no hay una sola descripción para formularlos; generalmente atañen a cuestiones de la vida cotidiana en sus diferentes aspectos, por ejemplo, problemas de producción, de transportación, de organización, comerciales, entre otros. En segundo término, se encuentran aquellos problemas planteados para demostrar y poner en práctica los métodos de solución de problemas y los algoritmos propuestos por los investigadores: se trata de entornos controlados, a menudo juegos o situaciones simplificadas de problemas reales.

Por otro lado, hay que señalar que, a menudo, los problemas reales han sido atacados mediante modelos que los reproducen de manera simplificada. La importancia de los modelos estriba entonces en que permiten centrarse en la solución evitando la interferencia de los detalles. Para llegar a tales modelos, se hace uso de una representación, que, valiéndose de la abstracción, elimina los detalles considerados irrelevantes para su solución. Las abstracciones han permitido llegar al desarrollo de formalismos, los cuales conceden un poder expresivo considerable, y han unido campos tan diversos del saber humano como el de los matemáticos y los artistas. Arte y ciencia, en este, sentido tienen, gracias a la geometría, un vínculo, un lenguaje, una manera de representar en abstracto, una vía para comunicar con imágenes. Para dar una idea clara del poder de

[2] Éstos son los problemas a los cuales la IA se ha orientado primordialmente.

la abstracción, se menciona a la geometría analítica como elemento unificador entre abstracción y concreción: álgebra + geometría. Gracias a estos formalismo es que se han sentado las bases para estudiar el conocimiento intuitivo y la experiencia cotidiana, por ejemplo, sabemos que un dibujo o una pintura en un espacio no euclidiano, no se crea conforme a la realidad que los ojos perciben, pues el ojo, ve sólo arcos y curvas, sino que es el intelecto el que permite interpretarlas y darles su "forma" final de líneas rectas [7].

De esta manera, con el uso de abstracciones para crear modelos, se tiene un entorno controlado de lo que sucede en el mundo real, y se puede plantear una solución, que posteriormente habrá de escalarse. Un ejemplo de tales modelos son las simulaciones de robots y sus ambientes, que posteriormente se extrapolan a los dispositivos robóticos reales.

Para cerrar esta sección, en la tabla 4.1 se mencionan algunos de los tipos de problemas de acuerdo con su procedimiento de solución.

Problema	**Descripción**	**Estrategia de Búsqueda**
Monoestado	Ambiente accesible. Se conoce el resultado de cada acción (a qué estado conducen las acciones). Hay estados únicos.	Búsqueda ciega y heurística
Multiestado	Ambiente inaccesible. Deben evaluarse los conjuntos de estados a los que se puede llegar. Cada estado inicial conocido se relaciona con una secuencia de acciones que garantizan lograr la meta.	Búsqueda ciega y heurística
De contingencia	No hay una secuencia de acciones que garanticen el logro de la meta. Hay que evaluar un árbol de acciones, no sólo una secuencia. Plantear la ejecución de acciones y luego revisar las contengencias derivadas de ellas.	Etapas alternadas de búsqueda y ejecución
De exploración	El *Ag* no sabe acerca del efecto de sus acciones ni conoce el ambiente. Vía la experimentacion, el *Ag* describe qué acciones conducen a qué estados.	Crea un mapa del ambiente para usarlo en futuras resoluciones de problemas

Tabla 4.1: Estrategias de búsqueda para diversos tipos de problemas y su descripción.

1.1. Formulación de un Problema

Como ya se comentó, el agente solucionador de problemas decide qué hacer para encontrar una solución (una meta), es decir, determina una secuencia de acciones conducentes a los estados deseables: aquellos que permiten alcanzar las objetivos del agente. De esta manera, un problema es toda aquella información que el agente necesita para actuar, y puede considerarse como un conjunto formado por estados E y acciones A:

Problema = {E, A}

Por tanto:

- El Ag_{obj} decide qué hacer para encontrar una solución (una meta m); determina una secuencia de acciones conducentes a los estados deseables (aquellos que permiten alcanzar las metas).
- El concepto de estado en la solución de un problema no es lo mismo que un estado del mundo real, pero un estado de la solución puede corresponder a un estado del mundo.
- Cada a es un procedimiento de transición entre estados ($e_i \xrightarrow{a} e_j$).

En este contexto, hay que entender la actividad de resolución de problemas como un procedimiento de búsqueda en espacio de estados, configurado por cuatro etapas:

1 Formular Metas: considera que una meta m es el último de un conjunto de estados E deseables del mundo para alcanzar el objetivo en turno del Ag (la meta general actual) vía la ejecución de cada $a \in A$.

2 Formular Problemas para lograr cada Meta: proceso para decidir el conjunto $\{A, E\}$ a considerar para alcanzar m. Los Ag desconocen cuál de las acciones será la mejor para alcanzar m (no son omniscientes, no están informados acerca del estado producido por cada una de sus acciones antes de ejecutarlas).

3 Buscar la solución: el algoritmo recibe como entrada la especificación P de un problema y regresa una solución S, que consiste en una secuencia de acciones:
$P \rightarrow$ [Algoritmo de búsqueda] $\rightarrow S$
S es la ruta de un e_o a un estado meta e_m, es decir el estado que satisface la prueba de meta τ
Solución óptima: es la de menor costo de ruta de entre (todas) las n soluciones halladas: $S^* = \arg\min K(S_i) | i = \{i, \ldots, n\}$

4 Ejecutarla: llevar a cabo las acciones propuestas en S (o S^*).

Definición Formal de un Problema

Def. 4 *Problema (P):* $P = \{E, \tau(e), \kappa(r)\}$

Donde:

- E: espacio de estados, aquellos alcanzables desde el estado inicial mediante cualquier secuencia de acciones.
- *Ruta* o *Camino*: secuencia r de estados conectados por una secuencia de acciones
  $r = \{e_i \xrightarrow{a_x} e_j \xrightarrow{a_y} e_k \ldots\}$ (e_i:$i-$ésimo estado).
- $\tau(e)$: función de prueba de meta, determina si un estado e es objetivo (el estado meta). Puede haber un conjunto de estados objetivo $O = \{e_1, \ldots, e_n\}$, $\tau(e)$ determina si $e \in O$
- $\kappa(r)$: función de costo de ruta.

Definición Formal de un Espacio de Estados

Def. 5 *Espacio de Estados (E):* $E = \{e_0, \varphi | f(e)\}$

Donde:

- e_0: estado inicial
- Las acciones pueden derivarse de cualquiera de estos elementos:
  - φ: operador que describe la transición entre estados: $\varphi : e_i \rightarrow e_j$
  - $f(e)$: función subsecuente. A partir de un estado devuelve el subsecuente: $f(e_i) = e_{i+1}$

Definición Formal de Costo de Ruta

Def. 6 *Función de costo de ruta ($\kappa(r)$):* $\kappa(r) = \sum_{i=1}^{n} c_i$

Donde:

- $c_i = c(e_x, a, e_y)$ y $c_i \in r$.
- $\kappa(r)$: función de costo de ruta. Asigna una ponderación numérica a cada camino o ruta r. Esta función puede reflejar la medida de rendimiento. Puede ser la suma de los costos individuales de cada acción emprendida a lo largo de r
- $c(e_i, a, e_j)$: costo de efectuar la acción individual a que va del estado e_i al estado e_j

De esta definición, se parte para realizar la búsqueda de la solución, como se ilustra en el siguiente esquema:

$$P \rightarrow [AlgoritmodeBúsqueda] \rightarrow S$$

La solución S es la ruta del estado inicial e_o al estado meta e_m, es decir el estado que satisface la prueba de meta. Por su parte, la solución óptima es la de menor costo de ruta de entre todas las soluciones, evaluado por la función $\kappa(r)$.

Es necesario resaltar que el conocimiento que el agente posea está relacionado con el tipo de soluciones que podrá alcanzar. En IA se han empleado diversas formas de de representar problemas; entre las más comunes están las siguientes:

- Lógica
- Estructura de Datos
- Grafos o gráficas

Se remite al lector al apéndice A para profundizar en primeros. Respecto a la última forma de representar, los grafos, por tratarse de una de las más relevantes para la búsqueda en espacio de estados, en la siguiente subsección se da una brevísima introducción a ellos con el fin de seguir el hilo de la exposición sobre la búsqueda, pero proporcionar los elementos básicos del entendimiento sobre grafos.

1.2. Grafos

Un grafo o gráfica, es una representación muy útil en IA, por la flexibilidad que reporta para ilustrar determinados datos y situaciones, y por que se ha desarrollado una teoría matemática muy sólida al respecto, que confiere herramientas dúctiles para su procesamiento y manejo.

Def. 7 *Grafo: Es un conjunto formado de nodos y arcos (también llamados vértices y aristas respectivamente):* $G = N, A$

Donde:

- N: conjunto no vacío de nodos.
- A: conjunto de arcos. Un arco representa la relación entre dos nodos pertenecientes a N, esto es, un par no ordenado de nodos.

Hay que resaltar los siguientes conceptos relacionados con los grafos:

- *Camino*: es una sucesión alternante de $n+1$ vértices (v) y n aristas (e) que comienza en algún nodo, por ejemplo v_0, y termina en otro, por ejemplo v_n: $(v_0, e_1, v_1, e_2, v_2, e_3, \ldots, v_{n-1}, e_n, v_n)$. Donde cada arista e_i es incidente en los vértices v_i y v_{i+1}.
- *Conectividad*: un grafo G es conexo, si dados dos vértices u y v de él, existe un camino de u a v.
- *Camino simple*: es un camino sin vértices repetidos.
- *Ciclo o circuito*: es un camino de longitud diferente de cero, sin aristas repetidas de un vértice v a v —a sí mismo.
- *Ciclo simple*: es un ciclo de v a v sin vértices repetidos —exceptuando el inicial y final, que son el mismo.

En la Figura 4.1 se muestra un grafo representado como un diagrama. Sin embargo, para efectos de su tratamiento computacional, se facilita el procesamiento utilizando otras representacione, por ejemplo, mediante una matriz de adyacencia (ver la figura 4.2), la cual utiliza una matriz cuyas dimensionas las integran los vértices ordenados. Cuando existe una arista incidente en dos vértices, se pone un 1 en el renglón y la columna correspondientes, y se pone un 0 en caso contrario. De manera rápida, se menciona que un grafo que carezca de ciclos, es decir, que sea acíclico, presentará, en primera instancia una diagonal principal con ceros, pues estas posiciones en la matriz corresponden a los mismos vértices; en caso contrario, indica que existen lazos, que son una forma de ciclos. De la misma forma, si se examinan las conectividades en los renglones-columnas de cada vértice, se puede seguir la sucesión que indiquen si existen ciclos (vértices repetidos).

Finalmente, se menciona que un grafo cuyos arcos definen un sentido, una dirección, se denomina *digrafo* o *grafo dirigido*, y ésta direccionalidad puede representar, por ejemplo, una relación de un nodo x a uno y, pero no de y a x.

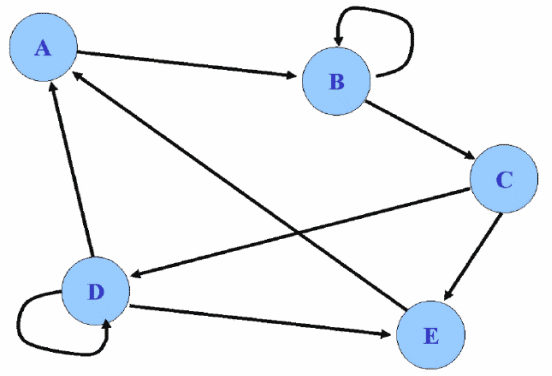


Figura 4.1: Ejemplo de un grafo representado como un diagrama.
$G = \{N, A\}$,
$G = \{\{A, B, C, D, E\}, \{(A, B), (A, D), (B, B), (B, C), (C, E), (C, D), (D, D), (D, E), (E, A)\}\}$

A	B	C	D	E	
A	0	1	0	0	0
B	0	1	1	0	0
C	0	0	0	1	1
D	1	0	0	1	0
E	1	0	0	0	0

Figura 4.2: Ejemplo del grafo de la Figura 4.1 representado mediante una matriz de adyacencia. La diagonal principal está indicada con celdas en amarillo. Cada presencia de arista se representa con un 1. Unos en la diagonal representan lazos del nodo respectivo.
$G = \{N, A\}$,
$G = \{\{A, B, C, D, E\}, \{(A, B), (A, D), (B, B), (B, C), (C, E), (C, D), (D, D), (D, E), (E, A)\}\}$

1.3. Representación de la búsqueda

Como ya se ha indicado, uno de los elementos esenciales para la búsqueda es el concepto de **Espacio de Búsqueda** o **Espacio de Estados**, que es una abstracción del problema, donde el ambiente se representa mediante variables (estados) básicas entre las cuales se halla la solución al problema —encontrada tras aplicar operaciones sobre los estados—. El espacio de estados puede representarse mediante diversos formalismos, entre ellos, se encuentran los grafos, las rejillas (*lattices*, en inglés), y más frecuentemente, un tipo especial de grafo: los **árboles**.

Un árbol es un tipo de grafo conexo y acíclico. Particularmente nos interesan los árboles con raíz o enraizados[3], los cuales se caracterizan de la siguiente manera:

Def. 8 *Árbol con raíz: Sea A un árbol con raíz, entonces se dice que A es un conjunto no vacío de elementos conocidos como nodos (n), tal que:*

- *A es un grafo conexo y acíclico*
- *Contiene un solo nodo inicial r, denominado raíz o nodo distinguido*
- *Los nodos restantes forman una colección ordenada de cero o más árboles disjuntos* $A_1, A_2, \ldots, A_m$

Es conveniente además, atender a las siguientes cuestiones:

- Nodo: es un punto discreto, una unidad de información (un dato o un registro de ellos, un estado del ambiente)
- Nodo etiquetado: Nodo al que se le asigna un valor determinado.
- Arista: Línea que une a un nodo con otro (llamado su sucesor).
- Camino: Secuencia consecutiva -sin ciclos- de aristas (y sus correspondientes nodos), por ej. $(v_0, v_1, \ldots v_n)$. De lo anterior se desprenden los siguientes hechos:
  - v_0 designa al nodo raíz
  - $v_0, \ldots, v_{n-1}$ son ancestros de v_n
  - v_n es un hijo o descendiente o sucesor de v_{n-1}
  - Si un nodo a es ancestro de otro b, entonces b es un sucesor de a
  - Si a y b son hijos de w, entonces a y b son hermanos
  - Si x no tiene hijos, se dice que es un nodo terminal u hoja
  - Si x no es un nodo hoja, entonces es un nodo interno o una rama
  - Subárbol es un conjunto de nodos y aristas cuyo nodo raíz es el descendiente de algún nodo interno de un árbol mayor que lo contiene

Hay que remarcar los siguientes conceptos:

- Hoja: nodo Terminal, sin sucesores.

[3]Por economía de lenguaje, utilizaremos en estos apuntes simplemente árbol para referirnos a un árbol enraizado, a menos de que se especifique algo contrario.

- Nodo solución: hoja que configura el estado que representa la solución de un problema.
- Rama: un camino que termina en una hoja del que pueden partir nuevos subárboles.
- Nivel: si al nodo raíz se le asigna el nivel 0, sus sucesores aumentarán en 1 el nivel. Así, el nivel de un nodo x, es la longitud del camino simple de la raíz a x.
- Profundidad: número máximo de nodos de una rama.
- Altura: máximo número de nivel de todos los nodos del árbol.
- Grado de un nodo: número de descendientes de un nodo.
- Grado del árbol: el máximo grado de todos los nodos del árbol.

A partir de la definición anterior y los conceptos relacionados, se menciona que los árboles codifican estructuras jerárquicas, relaciones lógicas, y nos ayudan a representar el espacio de estados de un problema atendiendo a estos lineamientos:

- Los nodos son estados del problema, pasos intermedios, soluciones probables.
- Las aristas son las transiciones entre nodos, transiciones logradas mediante la ejecución de las acciones posibles. También pueden representar las reglas para cambiar de estado.
- La raíz del árbol es el estado inicial, el punto de partida hacia la solución del problema.
- Los nodos hoja son las soluciones candidatas del problema.

Por último, se menciona que un problema puede resolverse mediante un proceso de descomposición recursiva del mismo. La raíz de esta perspectiva la encontramos si atendemos a lo dicho por Descartes [1,5]:

> Dividir cuantas dificultades se examinen, en cuantas partes sea posible y en cuantas se requiera para su mejor solución"

En este orden de ideas, y tomando en cuenta lo dicho anteriormente acerca de los problemas de transformación, es importante traer a cuenta el trabajo de Newell y Simon, el GPS (General Problem Solver), el Solucionador General de Problemas, el cual operaba mediante una heurística denominada Análisis de Medios y Fines, que se cimienta en el análisis y selección de las acciones que hay que realizar para conseguir un fin deseado. En palabras de Aristóteles [2]: hay que asumir los fines y ocuparse de los medios para conseguirlos. El análisis de medios y fines descansa sobre dos conceptos: *operadores* y *objetos*. Los primeros implican una transformación sobre objetos que produce objetos distintos. Por su parte, los objetos están descritos mediante sus características y sus diferencias mutuas, entre otras cosas. Del GPS se parte para identificar tres etapas básicas para solucionar problemas:

- Establecer la diferencia entre el estado actual y las posibles soluciones
- Seleccionar y aplicar el operador que reduzca o elimine tal diferencia
- Descomponer el problema en tantos subproblemas como requiera el operador

La búsqueda de una solución en un espacio de estados codificado como un árbol (al cual se le denomina árbol de búsqueda [4] cuya raíz es un nodo de búsqueda) tiene dos etapas fundamentales [2]:

- **Prueba de meta**: Identifica si el estado actual es un estado final (si se ha llegado a la solución)
- **Generación o Expansión de nodos**: Consiste en la aplicación de los operadores a un estado, para generar nuevos estados, es decir, expandir el nodo actual para obtener nodos sucesores (subárboles) en camino hacia la solución.

1.4. Una travesía en carretera

A continuación se muestra un ejemplo clásico para mostrar esta cuestión [2]. Dado el mapa carretero de la Figura 4.3, nos formulamos la siguiente tarea:

> Ir de la Ciudad de México (D.F.) a alguna de las ciudades a las que está unido mediante una carretera

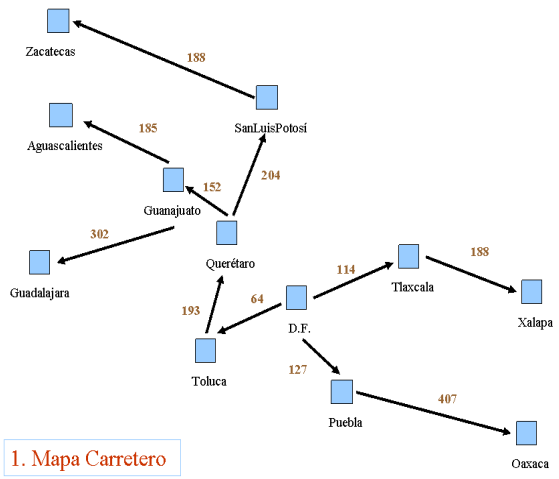


Figura 4.3: Mapa carretero de conectividades y distancias entre ciudades del centro de México.

Esto plantea un problema representable mediante un árbol de búsqueda. En el árbol, cada nodo representa una ciudad, y cada arista una carretera. Hay que

[4] Un árbol de búsqueda es un árbol enraizado, por ello, al hablar de "árboles" simplemente, se entenderá que se hace alusión a los árboles de búsqueda.

recordar que las carreteras por lo regular tienen dos sentidos; sin embargo, en el caso que nos ocupa, tal posibilidad depende de los datos del problema que se esté analizando. En este ejemplo, sólo se establecen las direccionalidades que aparecen en el mapa carretero sin considerar más sentido que el expresado por las flechas.

Regresando a la representación del problema —nuestro árbol de búsqueda—, el nodo raíz es la ciudad origen (D.F.)[5] y el nodo destino, puede ser alguna de las ciudades restantes. Las ramas del árbol están integradas con las conexiones que haya entre las carreteras, como se muestra en la figura 4.4.

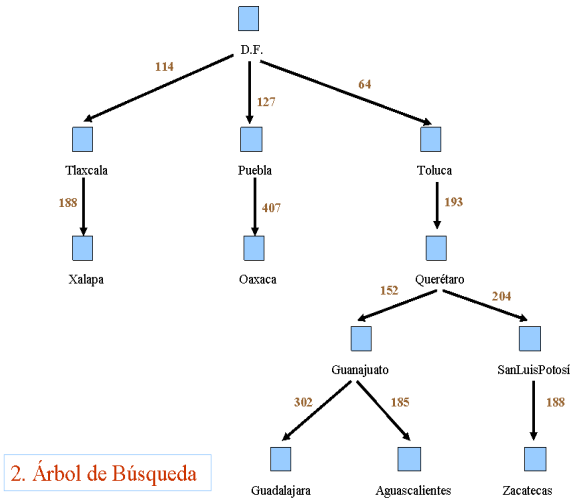


Figura 4.4: Representacion del árbol de búsqueda que tiene como origen la Ciudad de México (D.F.) y destino alguna de las ciudades conexas del mapa carretero.

De acuerdo con lo que se ha mencionado hasta el momento sobre la formulación y resolución de problemas y la representación del espacio de búsqueda, a continuación se revisan algunos de los procedimientos (los más elementales) que se han desarrollado para intentar resolver la cuestión de la resolución de problemas como una búsqueda.

2. Búsqueda ciega o desinformada

En este apartado se muestra la manera en que se puede realizar la búsqueda sobre un espacio representado como un árbol. Para ejemplificar mejor las búsquedas, se toman los siguientes datos: en primer lugar, un Mapa Carretero Extendido –con más conexiones que las que se mostraban originalmente en el primer mapa, como se observa en la figura 4.5.

Partiendo de lo anterior, tenemos, en primer término, la **búsqueda ciega o desinformada**, la cual se orienta a encontrar la solución siguiendo una exploración sistemática del espacio de estados –el árbol– mediante un proceso de

[5]Se le conoció así a la Ciudad de México hasta el año 2016, dispense el lector este anacronismo. Ver el decreto que así lo dispuso en [37]

ensayo y error. Tal exploración tiene dos versiones: **profundidad** y **amplitud**, de acuerdo con el orden en que va revisando los nodos, esto es, el orden en que va generando o expandiendo los nodos para aplicarles la prueba de meta. Veamos a continuación en qué consisten estos procedimientos.

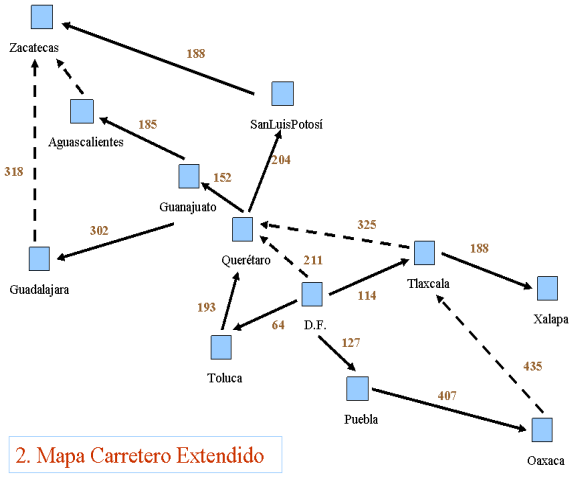


Figura 4.5: Mapa carretero extendido de algunas ciudades de México. Las nuevas carreteras se muestran con líneas punteadas.

Consecuentemente, se obtiene un árbol con mayores ramificaciones, según se ilustra en la Figura 4.6.

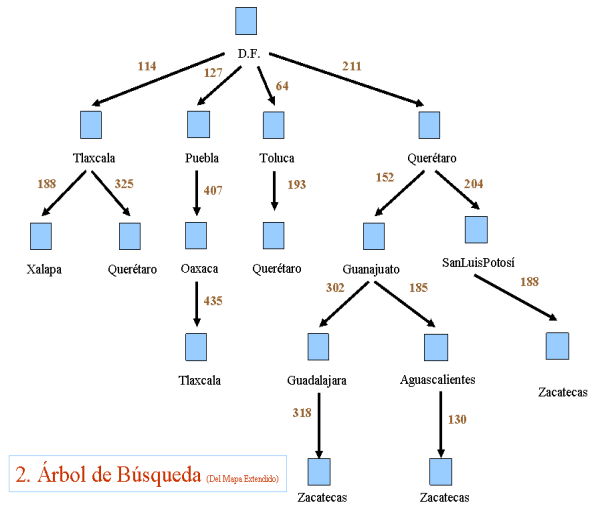


Figura 4.6: Árbol de Búsqueda –abreviado- resultante del mapa carretero extendido.

2.1. Búsqueda por profundidad

Realiza la expansión de nodos siguiendo todos los niveles del árbol, rama por rama, comenzando desde el nodo raíz hasta terminar con el nodo hoja de cada una de ellas. En las Figuras 4.7 y 4.8 se ilustra cómo se lleva a cabo esta expansión.

En cada nodo se aplica la prueba de meta, en caso de que ésta falle, continúa la búsqueda de dos maneras posibles:

a) El nodo tiene sucesores: se comienza la búsqueda con su primer sucesor (descenso en profundidad).

b) El nodo no tiene sucesores: se retrocede al nodo padre y se buscan nuevos sucesores (retroceso o *backtracking*).

Los requisitos de memoria son mínimos, pues lo único que se requiere almacenar es la ruta en cuestión —el camino desde la raíz hasta un nodo hoja— junto con los nodos hermanos inexplorados del nivel actual. Esto puede hacerse con el uso de una estructura de datos de tipo pila. Este procedimiento de búsqueda puede instrumentarse de manera recursiva, llamándose sucesivamente en cada nodo hijo.

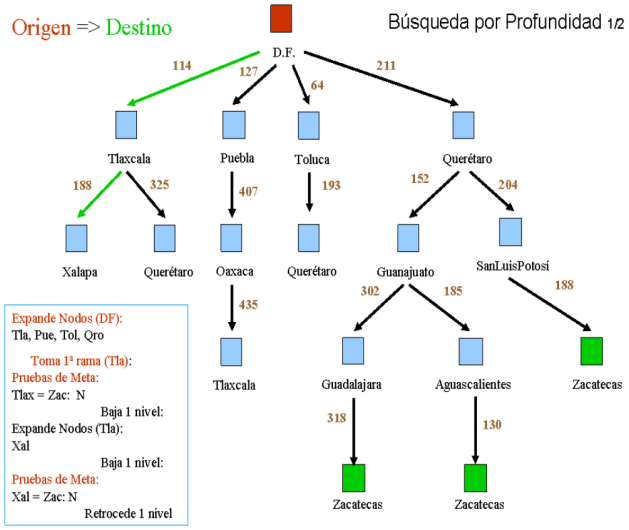


Figura 4.7: La rama marcada en verde es la primera que se expande, aplicando la prueba de meta a todos los nodos que la conforman, nivel por nivel.

Si un espacio de estados tiene un **factor de ramificación** b y una **profundidad máxima** m, se requerirán bm nodos, esto es una complejidad espacial $O(bm)$. En tanto que la complejidad temporal de esta búsqueda es $O(bm)$.

Se pueden mencionar dos desventajas de la búsqueda en profundidad. La primera es que se comience a avanzar por una ruta equivocada que no lleva a la solución, pues hasta que no esté completamente recorrida ésta, no será posible retomar una rama adecuada en el árbol. En segundo lugar y de manera preponderante, hay que observar que la gran desventaja se presenta cuando el árbol de búsqueda es muy profundo. Resulta evidente que en un árbol infinito,

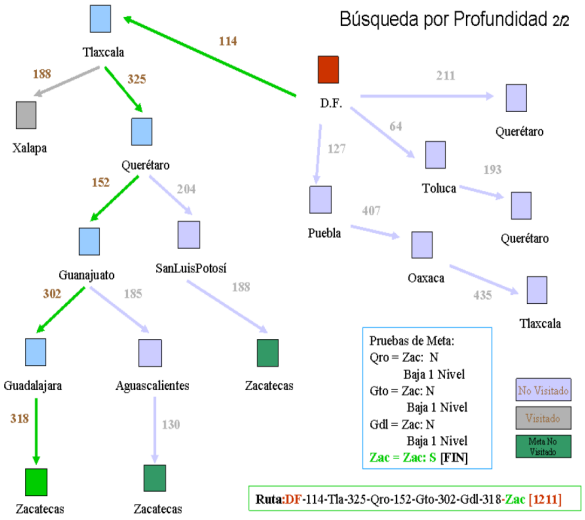


Figura 4.8: A continuación se expande la rama contigua (hay retroceso), haciendo las pruebas de meta a todos los nodos sucesores hasta que encuentra el nodo destino o meta: Zacatecas, de modo que termina la búsqueda.

ésta búsqueda no encontraría una ruta adecuada, pues no sería posible aplicar el retroceso ya que no se terminaría de recorrer la rama seleccionada.

Note el lector que la búsqueda por profundidad no es completa ni óptima [2].

2.2. Búsqueda por amplitud

La búsqueda por amplitud realiza la expansión de nodos nivel por nivel, rama por rama. Primero se expande el nodo raíz, es decir, se obtienen sus sucesores, y se les aplica la prueba de meta, y así sucesivamente, nivel por nivel, en el orden de los sucesores. En otras palabras, examina las rutas de longitud 1 —indicada por el nivel del árbol—, luego las de longitud 2, etc.

Es importante señalar que, si existe una solución, la búsqueda por amplitud la encontrará y, si existen varias soluciones se hallará la más próxima (con respecto al nodo del árbol) [2]. Esta estrategia se puede instrumentar con el uso de dos estructuras de datos: una pila y una cola. En ésta última se almacenan los nodos hermanos que se evaluarán por la prueba de meta, esto es, los nodos sucesores se ponen al final de la cola a medida que se van expandiendo y en ese orden se van analizando. De tal manera, los nodos de menor nivel se expanden antes que los nodos más profundos.

Contrariamente a la búsqueda por profundidad, los requerimientos de memoria de la búsqueda por amplitud son serios: en el caso de expandir todos los nodos —salvo el útimo nodo, que sería la solución– se tiene una complejidad espacial de $O(bd+1)$. Siendo b los sucesores de un nodo y d la profundidad del árbol. Lo mismo podemos decir de la complejidad temporal: $O(bd+1)$, sin embargo, es importante considerar que las necesidades de almacenamiento son más serias que las de tiempo de ejecución.

En las figuras 4.9- 4.15 se ilustra esta búsqueda.

En contraste con la búsqueda por profundidad, podemos decir que la búsqueda por amplitud es óptima para el caso en que los costos de cada arista del árbol sean los mismos (costo uniforme para pasar de una estado a otro) [6]. Por otro lado, la búsqueda por amplitud es completa: la solución se encuentra si d y b son finitos.

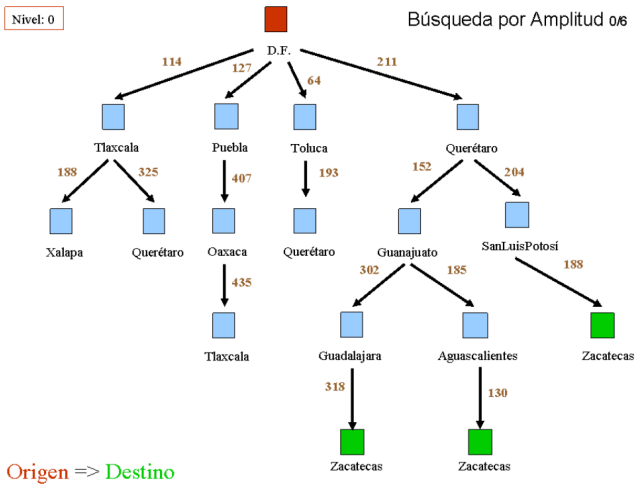


Figura 4.9: El árbol de búsqueda para la ruta D.F. – Zacatecas siguiendo el Mapa Carretero Extendido.

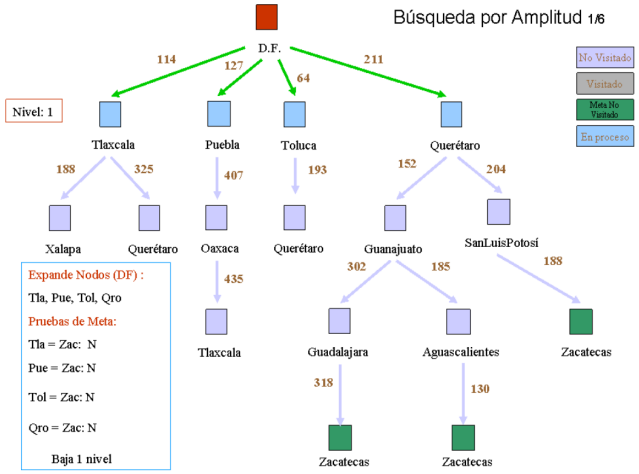


Figura 4.10: Expansión del primer nivel (los sucesores del nodo raíz), y su aplicación de la prueba de meta.

[6] Recordar que el costo de ruta, expresado como la suma de los costos de cada arista a lo largo del camino raíz-nodo solución, como quedó expresado en la sección de Formulación de problemas.

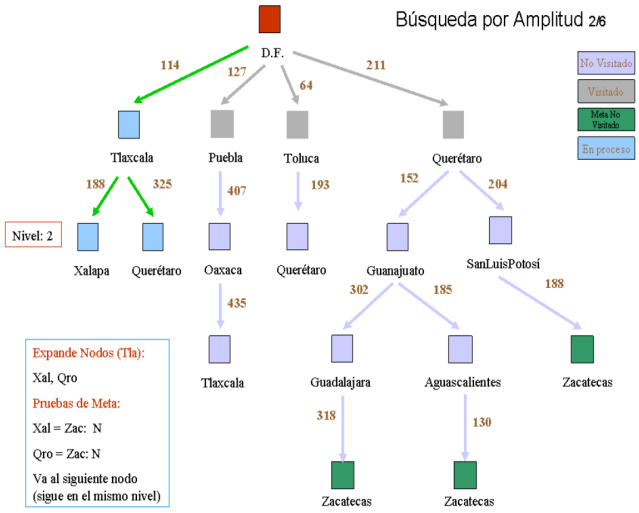


Figura 4.11: Como aún no se halla la solución, se expande el segundo nivel, a partir de la primera rama.

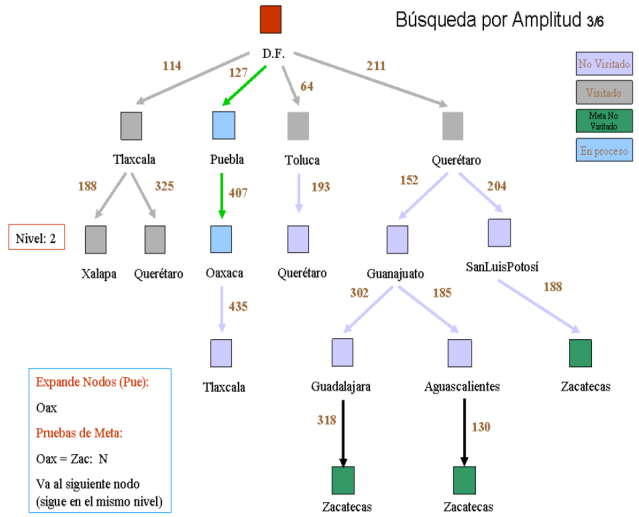


Figura 4.12: Se sigue expandiendo y aplicando la prueba de meta con todos los nodos del segundo nivel.

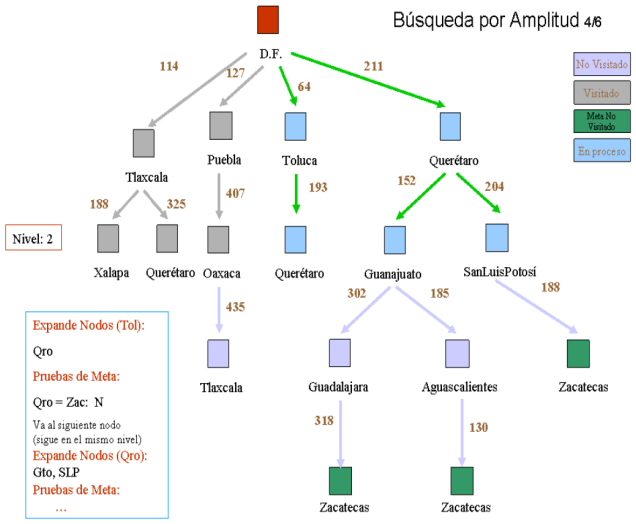


Figura 4.13: De la mima manera se continúa con los nodos del mismo nivel de las ramas siguientes.

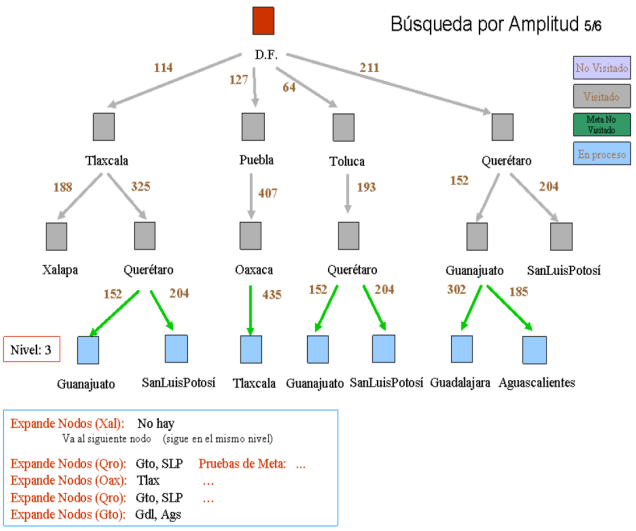


Figura 4.14: Debido a que no se ha hallado la solución, se expande y revisa el siguiente nivel del árbol.

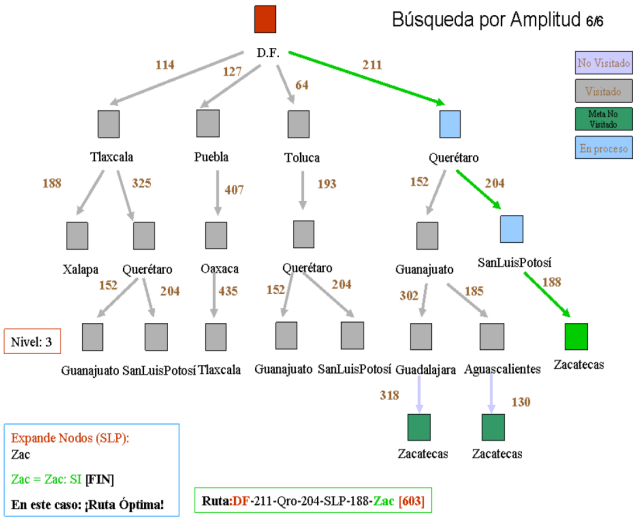


Figura 4.15: Finalmente se halla una hoja meta y termina la búsqueda. La rama verde representa la solución.

2.3. Ejemplo

El programa que implementa la búsqueda en profundidad se muestra en el listado 4.1:

```
# -*- coding: utf-8 -*-
# primeroProfundidad.py
# Implementa la busqueda primero en profundidad. Esta adaptado para el mapa carretero de las ciudades del centro de la Cd Mx (DF)
# Autor:
# Dr Wulfrano Arturo Luna Ramirez
# Referencia:
# Shivali Bhadaniya (Dic 21 202):
# https://favtutor.com/blogs/depth-first-search-python
#

# Diccionarios Python para representar el grafo como una lista de adyacencia
mapaCarretero = {
  'DF' : ['Tlaxcala','Puebla','Toluca','Queretaro'],
  'Tlaxcala' : ['Xalapa','Queretaro'],
  'Puebla' : ['Oaxaca','Tlaxcala'],
  'Toluca' : ['Queretaro'],
  'Oaxaca' : ['Tlaxcala'],
  'Toluca' : ['Queretaro'],
  'Queretaro' : ['Guanajuato','SanLuisPotosi'],
  'Guanajuato' : ['Guadalajara','Aguascalientes'],
  'Guadalajara' : ['Zacatecas'],
  'Aguascalientes' : ['Zacatecas'],
  'SanLuisPotosi' : ['Zacatecas'],
  'Xalapa' : [],
  'Zacatecas' : []
      }
#
# Funciones para el mapa carretero
```

```
#

# Regresa el resultado de la comparaci n
# if origen == destino:
def esMeta(a,b):
  return (a == b)

# Regresa los nodos conectados a nodo en el grafo
def expandir(nodo,grafo):
  return grafo[nodo]

# Regresa la elecci n del usuario en cuanto a origen/destino
def origenDestino():
  origen = input("Origen?: ")
  destino = input("Destino?: ")
  return (origen,destino)

# Realiza la B squeda Primero en Profundidad
def bpp(visitado, grafo, origen, destino,meta):
  # Prueba de meta
  if esMeta(origen,destino):
    ruta.append(origen)
    return 1
  # Revisar mientras no se haya llegado a la meta
  elif meta == 0 or meta == 2:
    hijos = expandir(origen,grafo)
    # La rama no es solucion
    if hijos == []:
      ruta.remove(origen)
      return 2
    elif origen not in visitado:
      visitado.add(origen)
      ruta.append(origen)
      for hermano in grafo[origen]:
        ruta.append(hermano)
        # Llamada recursiva para recorrer la rama
      meta = bpp(visitado, grafo, hermano,destino,meta)
        # Caso donde la rama es soluci n
      if meta == 1:
        ruta.remove(hermano)
        return 1
      ruta.remove(origen)
    # Caso donde la rama no es soluci n
    if meta == 2:
      if origen in visitado:
        ruta.remove(origen)
        return 2
  return 0
#
# Realiza el funcionamiento de b squeda en el mapa carretero
#
def bppMapa(origen,destino):
   bpp(visitado, mapaCarretero, origen, destino,meta)

# Arbol de b squeda:
#       DF
#    /    / \    \
#      Tla   Pue   Tol    Qro
#     /  \    | |   /  \
#       Xal   Qro  Oax   Qro  Gto  SLP
#    /  \   |   |   | \   \
#      Gto  SLP Tla ... Gdl Ags Zac
```

```
#     / \ \  |      | |
#    Gdl  Ags Zac ...     Zac Zac
#     |  |
#    Zac   Zac
#
#
meta = 0
ruta = []
visitado = set()

# Captura el origen y el destino preferidos por el usuario
(origen,destino) = origenDestino()
# Invoca la b squeda en el mapa carretero
bppMapa(origen,destino)
print(ruta)
```

Código 4.1: Código del agente que busca la mejor ruta en para viajar por el centro de México. `primeroProfundidad.py`

Un ejemplos de la corrida de este programa se muestra a continuación:

```
>python3 primeroProfundidad.py
Origen?: 'DF'
Destino?: 'Zacatecas'
Esto da como resultado:
origen -> destino
'DF' ->'Zacatecas'
 Se invoca:
bppMapa(origen,destino)
 Es decir:
bppMapa('DF','Zacatecas')

 Salida esperada (primera rama-solución en profundidad):

DF
 /
 Tla
\
Qro
 /
 Gto
/
 Gdl
|
 Zac

 Ruta:
'DF'-> 'Tlaxcala'->'Queretaro'->'Guanajuato'->'Guadalajara'->'Zacatecas'
```

3. Búsqueda heurística o informada

Se denomina *búsqueda informada* a aquel procedimiento de búsqueda que incorpora conocimiento específico del dominio de aplicación. El resultado de esta

adición es una mejora en la eficiencia de la búsqueda. El conocimiento, en este caso, se representa mediante una **función de evaluación**. Dicha función permite obtener un estimado de lo deseable que es la expansión —y posterior exploración— de los nodos. A continuación se muestra el caso más común del uso de esta función: la búsqueda voraz primero el mejor [2].

3.1. Búsqueda por lo más prometedor

El nombre de este procedimiento de búsqueda atiende al hecho de que se va expandiendo aquel que tenga la mejor puntuación según la función de evaluación $f(n)$, y el proceso continúa expandiendo los nodos hijos mejor calificados sucesivamente hasta encontrar la meta. Es evidente que escoger el camino que nos marca el nodo mejor calificado en cada paso no siempre conduce a la mejor ruta; en algunos casos, esta búsqueda no prosperará y asumir una función de evaluación dejará de ser una ventaja tal.

Operativamente, la búsqueda por lo más prometedor (por lo que parece mejor) se puede implementar basándose en una búsqueda por amplitud agregando la función de evaluación.

Como se puede ver, hay toda una gama potencial de funciones de evaluación; por lo general, se utiliza una medida estimada a partir del costo de la solución y se intenta reducir al mínimo dicho costo.

Aquí conviene mencionar una tipo de búsqueda que se basa en la minimización del costo de ruta: la búsqueda de costo uniforme, la cual expande siempre el nodo con el costo de ruta ($g(n)$) menor. Bajo esta perspectiva, se puede decir que la búsqueda en amplitud es una búsqueda de costo uniforme, donde el costo de ruta es igual a la profundidad: $g(n) = profundidad(n)$. Con esto en mente, conviene aclarar que la búsqueda por lo más prometedor realiza una exploración dirigida a la meta, expandiendo los nodos más cercanos a ella. Por tal motivo, $f(n)$ debe contener algún cálculo del costo de ruta de un estado inicial a uno más cercano a la meta.

Esto se ve en la ecuación 3.1:

$$f(n) = h(n) \tag{4.1}$$

Donde $h(n)$ es una función heurística [7] que incorpora conocimiento del dominio para la solución: costo estimado del camino más barato desde el nodo n a un nodo objetivo. Cualquier función puede utilizarse como heurística, con tal de que cumpla con lo expresado en la ecuación 3.1:

$$h(n) = 0, sin = nodometa \tag{4.2}$$

Una heurística es un procedimiento falible que ayuda a encontrar la solución de un problema. Debido a esta falibilidad y a que no se presta para su explicación, se considera a un procedimiento heurístico como opuesto a algorítmico. Concretamente, en el caso de la búsqueda, una función heurística es aquella de la cual se obtiene un estimado del costo de la solución. Para el ejemplo que se

[7] Heurística proviene de *eureka*, del griego *heuriskein*, encontrar o descubrir. Es común la afirmación que atribuye a Arquímedes el salir corriendo desnudo de una tina de baño gritando "¡eureka!, ¡eureka" a propósito del principio de flotación [2].

ha venido usando, $h(n)$ podría representar la distancia en línea recta entre las ciudades y no la distancia en kilómetros que tienen las carreteras.

Esta búsqueda no es óptima, pues la heurística puede fallar, al tomar una ruta que parece más prometedora, pero que termina no siéndolo en realidad. Tampoco resulta completa, porque puede ir a probar una ruta infinita sin regresar nunca a probar otras posibilidades.

Tiene una complejidad temporal (y espacial) $O(b_m)$, con m igual a la profundidad máxima del espacio de búsqueda. Sin embargo, hay que notar que una función heurística bien planteada puede disminuir considerablemente esta complejidad.

3.2. Búsqueda heurística A*

Esta búsqueda trata de paliar las desventajas anteriormente mencionadas. Este procedimiento conjunta la búsqueda preferente por lo más prometedor y la búsqueda de costo uniforme. Es decir, combina las dos funciones de evaluación [2]:

$$f(n) = g(n) + h(n) \tag{4.3}$$

Donde: $f(n)$ = costo de la solución más barata.

Esta estrategia es razonable puesto que, si se intenta encontrar la solución más barata, conviene entonces analizar el nodo cuyo valor de f sea el menor. La relevancia de A^*, aparte, de ser razonable, es que, bajo una condición, es completa y óptima. La condición es que h sea una heurística admisible. Esto significa que nunca sobreestima el costo de alcanzar la meta. Se dice que una heurística así es optimista pues plantea un costo de solución menor al costo real.

De tal manera, g representa el costo exacto de la solución y h nunca lo sobreestima. Una heurística admisible es también una heurística consistente o monotónica, si, para cada nodo n y cada sucesor n' de n generado por cualquier acción a, el costo estimado de alcanzar el objetivo de n no es mayor que el costo de alcanzar n' sumado al costo estimado ofrecido por n: $h(n) \leq c(n, a, n') + h(n')$.

A continuación se reiteran las características de A^* [2]:

- Es óptima siempre que h sea consistente.
- Si $h(n)$ es consistente, los valores de $f(n)$,a lo largo de cualquier camino no disminuyen. Esto plantea la posibilidad de crear contornos o curvas de nivel en el espacio de estados. Así, los nodos explorados con heurísticas adecuadas se van tomando conforme a estas curvas, las cuales van haciéndose concéntricas en torno a la ruta óptima. De lo anterior se desprenden dos afirmaciones:
  - A^* expande todos los nodos en donde $f(n) < f*$, donde $f*$ es el costo de ruta de la solución óptima.
  - A^* puede expandir algunos nodos en el contorno meta, para el que $f(n) = f*$, antes de proceder a seleccionar el nodo meta.
- A^* es óptima, pues la primera solución encontrada es la óptima.
- A^* es completa.

- A^* es óptimamente eficiente para cualquier función heurística. Así, ningún otro algoritmo óptimo garantiza expandir menos nodos que A^* pues si se expanden nodos que no cumplan con $f(n) < f*$ puede no incluir la solución óptima.

Desafortunadamente, hay que señalar que para la mayoría de los problemas la cantidad de nodos dentro del contorno meta ¡es exponencial!, por lo que el tiempo de búsqueda de A^* requiere un tiempo de cómputo inaceptable. Lo mismo sucede con el espacio necesario, que a menudo, hace que A^* se quede sin memoria antes de que agote su tiempo. Ante esto, se menciona que resulta poco práctico insistir en la optimalidad de la solución, por lo que hay que considerar el uso de algoritmos subóptimos, o bien, variantes de A^* con memoria acotada.

Finalmente, se apunta un pequeño comentario sobre las funciones heurísticas: resulta interesante la manera en que estos procedimientos ayudan para lograr una mejora relevante para el problema de búsqueda, pero ¿cómo se crean las heurísticas? Una respuesta la ofrece el análisis de un problema relajado, el cual impone menos restricciones a los operadores del problema original; pues es frecuente que el costo de la solución en este caso constituya una buena heurística para el problema original, pues la solución óptima del problema original es también una solución en el problema relajado y debe ser al menos tan cara como la solución óptima del problema relajado [2]. Además, se menciona que, si el problema se especifica en términos formales, es posible construir automáticamente el problema. A partir de estas definiciones se crean heurísticas automáticamente. También pueden crearse heurísticas para subproblemas y, por composición, obtener la heurística del problema completo. Una manera de encontrar heurísticas automáticamente es aplicar procedimientos de Aprendizaje Automático, concretamente, el Aprendizaje Inductivo, donde cada estado, junto con el costo real de la solución a partir de tal punto, corresponde a un ejemplo para el algoritmo, que va colectándolos para construir la heurística.

En la figura 4.16 se muestra un resumen de los algoritmos de búsqueda en espacio de estados que se han revisado en este capítulo, como la manera de conseguir la solución de un problema enunciada al principio de este capítulo.

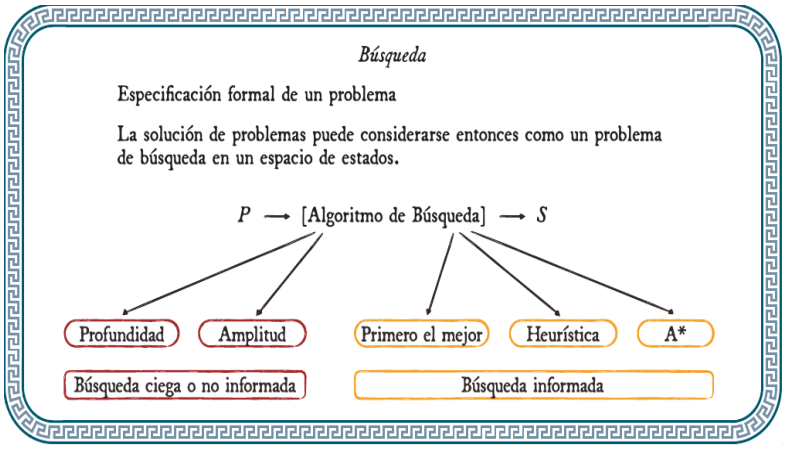


Figura 4.16: Algoritmos de búsqueda informada y no informada en un espacio de estados.

Para terminar, se reitera que, de entre estos algoritmos, A^* es el que está situado como mejor candidato de la búsqueda informada, bajo ciertas condiciones. Esto se ilustra en la Figura 4.17.

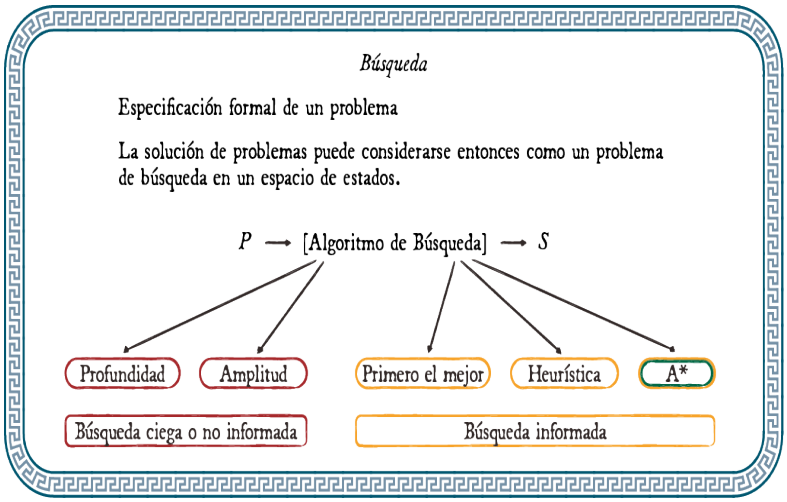


Figura 4.17: Solucionar un problema mediante algoritmos de búsqueda en espacio de estados: A^* como la mejor propuesta.

4. Comentario final

Como última nota, hay que señalar la diferencia entre la manera de resolver problemas mediante búsqueda y la resolución algorítmica [8]. Un algoritmo es una secuencia ordenada y finita de instrucciones que conduce de manera consistente a la solución de un problema. El algoritmo precisa recibir una entrada de datos, a la cual se aplique una serie de instrucciones, y se obtienen datos como salida, en este caso, la solución del problema —si el algoritmo es capaz de obtenerla–. Existen tres posibilidades al plantear soluciones algorítmicas puras [8]:

- Obtener una solución en un tiempo finito.
- Que la solución no se logre en un tiempo finito.
- Que el algoritmo no sea adecuado para el problema planteado.

Los procedimientos de búsqueda descritos anteriormente, principalmente la búsqueda heurística, intentan resolver estas dificultades, sobre todo las dos últimas, pero hay que tomar en cuenta que ambas visiones, la algorítmica y la búsqueda heurística son complementarias, puesto que una heurística puede convertirse en un algoritmo y de un algoritmo pueden derivarse heurísticas aceptables para un problema particular [8] y con ello, disminuir el espacio de estados y por ende, el tiempo para alcanzar la solución; además de que plantean una mejora, una guía para la operación del algoritmo dentro de un dominio determinado.

5. Resumen

Agentes con objetivos: el caso de la búsqueda

En este capítulo se abordó el problema de búsqueda en agentes racionales

- Muchos problemas pueden resolverse mediante su representación en un espacio de estados.
- Los agentes racionales pueden usar la búsqueda en espacio de estados para resolver sus tareas.
- Existen diversos métodos para abordar la búsqueda en espacio de estados, en general, se dividen en procedimientos informados y no informados, que utilizan enfoques y algoritmos de exploración, como la búsqueda en profundidad o amplitud, y la incorporación de heurísticas.
- En este capítulo se proporcionó un ejemplo práctico de búsqueda para ser incorporado directamente como una de las habilidades de un agente basado en objetivos.

6. Actividades de aprendizaje

6.1. Ejercicio de programación

- Objetivo: desarrollar el Agente Viajero, un Ag_{obj} capaz de viajar por el centro del país, por ejemplo:

  > ir de la Ciudad de México (D. F.) a alguna de las ciudades a las que está unido mediante una carretera

- Desarrollo: se debe plantear un agente que incorpore el algoritmo de búsqueda primero el mejor, descrito en el capítulo.
  - Realizar el análisis del Entorno de Trabajo (RAES)
  - También se debe modificar el programa de búsqueda mostrado en el listado 4.1, y realizar las siguientes tareas:
    - Incorporar el algoritmo de búsqueda a un programa de agente (como alguno de los agentes Triángulo o Leandro), es decir, incluirlo como parte del ciclo *percibir-decidir-incidir*, y denominarlo `agenteViajero-V0.py`.
    - Utilizar el algoritmo de *búsqueda por profundidad*. Esto es, cambiar el algoritmo que el ejemplo original implementa e incorporarlo como opción para que un usuario decida qué búsqueda implementar (`agenteViajero-V1.py`)
    - Modificarlo una vez más (`agenteViajero-V2.py`) para utilizar el algoritmo $A*$ con las siguientes definiciones, a partir de la ecuación 3.2:
      $f(n) = g(n) + h(n)$
      Donde:
      $g(n) = distancia\ en\ km\ de\ una\ ciudad\ a\ otra \in \mathbb{R}$
      $h(n) = interés\ turístico \in \mathbb{R}$
    - Para ello puede extenderse la tabla de datos del ejemplo del mapa carretero con valores que implementen las medidas para g y h.

6.2. Cuestionario 4

Responda las siguientes preguntas sobre búsqueda en espacio de estados [8].

1. ¿Qué representa el estado inicial en una búsqueda en espacio de estados?
   a) El estado final deseado
   b) El estado actual del agente
   c) Un estado intermedio cualquiera
   d) El estado desde donde comienza la búsqueda
2. El estado inicial en una búsqueda en espacio de estados ...
   a) No requiere ser definido

[8] Los cuestionarios de este libro fueron realizados parcialmente con una IA Generativa. Ver el esquema de trabajo propuesto en el apéndice C.

b) Corresponde al nodo raíz del árbol
c) Se genera aleatoriamente
d) Es el objetivo a alcanzar

3. Para comenzar una búsqueda en espacio de estados se necesita ...

a) Conocer posibles estados finales
b) Definir un estado inicial
c) Tener claros los operadores posibles
d) Calcular el costo de cada acción

4. En una búsqueda en espacio de estados, el estado final es ...

a) Un estado intermedio cualquiera
b) El objetivo que se busca alcanzar
c) El estado actual del agente
d) El nodo raíz del árbol

5. ¿Qué representa el estado final en una búsqueda en espacio de estados?

a) El estado inicial
b) El objetivo que se busca alcanzar
c) Un estado sin expandir aún
d) El nodo hoja del árbol

6. Durante la búsqueda en espacio de estados se debe ...

a) Partir de un estado final conocido
b) Verificar si un estado es el final
c) Evitar definir un estado final claro
d) Desconocer el estado final buscado

7. Los operadores en una búsqueda en espacio de estados sirven para ...

a) Establecer el orden de expansión
b) Generar nuevos estados sucesor
c) Definir el estado inicial
d) Determinar el estado final

8. ¿Qué representan los operadores en una búsqueda en espacio de estados?

a) Los nodos hoja del árbol de búsqueda
b) Las acciones aplicables a cada estado
c) Los estados finales posibles
d) Los estados sin expandir aún

9. Durante la búsqueda en espacio de estados se aplican ...

a) Heurísticas para podar el árbol
b) Operadores para generar nuevos estados
c) Reglas para determinar el estado final
d) Métricas para medir el costo total

Capítulo 5

Sistemas Multi-Agente y Modelación

Objetivos

- Conocer los fundamentos teóricos y la definición de los Sistemas Multi-Agente y la Simulación/Modelación Basada en Agentes, así como su utilidad y aplicaciones
- Identificar las principales características de los Sistemas Multi-Agente y la Simulaciones/Modelación Basada en Agentes y los fundamentos de su desarrollo
- Programar un ejemplo en una plataforma de software orientado a multi-agentes: NetLogo

Id conmigo a la fábrica-ciudad: venid, que quiero
contemplar con los pueblos las creaciones violentas,
la gestación del aire y el parto del acero,
el hijo de las manos y de las herramientas.
Miguel Hernández, *La fábrica-ciudad*

1. Sistemas Multi-Agente:

Los agentes, como sistemas computacionales (de *software* o *hardware*) que están situados en un ambiente y cuya operación busca cumplir con un objetivo de diseño, tienen ciertas características ya abordadas en capítulos anteriores, como la *proactividad* (desenvolverse de forma persistente en el logro de sus objetivos), la *deliberación* (se capaz de razonar sobre sus percepciones y estímulos recibidos) y, particularmente, su *capacidad social* (la comunicación interagente y también con intersistema, es decir, con otros sistemas, sean o no agentes).

La última característica permite crear ensambles o conglomerados de agentes, conocidos como Sistemas Multi-Agente (SMA), que comparten su ambiente e interactúan para conseguir objetivos comunes, lo cual les confiere una capacidad de tolerancia a fallos al tener un control descentralizado. Esto los hace ideales para enfrentar distintas tareas con un alto grado de robustez.

Para adentrarnos en nuestro estudio partamos de esta definición de SMA.

Def. 9 *Sistemas Multi-Agente (SMA)Un SMA es un grupo de agentes (homogéneos o heterogéneos) que interactúan entre ellos (cooperativa o competitivamente) dentro de un ambiente en particular y comparten recursos, estímulos y posiblemente un objetivo común.*

Los SMA han sido utilizados extensamente para el desarrollo de aplicaciones en distintos dominios incluyendo el comercio electrónico y sus aplicaciones industriales, hasta simulaciones de fenómenos sociales y naturales, como aquellas de desastres, además de apoyo a tareas de salvamento y recuperación postevento.

Las ideas clave detrás de un SMA se pueden enlistar como sigue [38]:

- La existencia de problemas complejos distribuidos e inasibles para agentes individuales, ya sea por la carencia de conocimiento o capacidades
- No hay un control global de todo el sistema, es decir, su operación no es controlada centralmente.
- La información y datos que se utilizan son distribuibles apropiadamente.
- Hay una forma (o varias) de coordinación interagente.

En las siguientes secciones se tratan las características más relevantes de los SMA y se introducen formas de programarlos.

1.1. Agentes sociales

Como ya se indicó, una característica importante de los *Ag* es su sociabilidad: que es la habilidad para interactuar con otros agentes (posiblemente humanos)

mediante algún lenguaje de comunicación interagente y cooperar o competir con ellos.

Esto es:

> Dada su habilidad social, los agentes pueden conformar ensambles (homogéneos o heterogéneos) que al estar imbuidos en un ambiente común, se relacionan entre sí por cooperación o competencia, para el logro de un objetivo común.

En la figura 5.1 se puede observar una representación de varios agentes formiciformes que trabajan como obreras en una tarea de forrajeo representados por los círculos rojos.

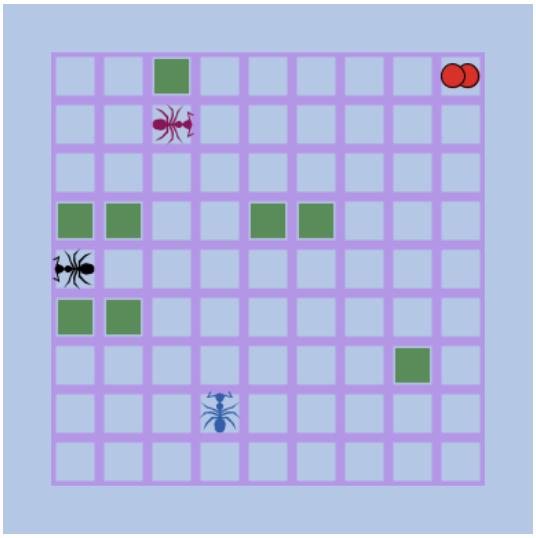


Figura 5.1: Distintos agentes operando en un mismo ambiente para el logro de un objetivo en común.

Para tener una noción más clara de los SMA, veamos una definición un poco más formal:

Def. 10 *Sistemas Multi-Agente (SMA):* $SMA = \{a_1^t, a_2^t \ldots, a_N^t\}, t \in T$
$\exists a^x, b^y \mid (a, b \in MAS, a \neq b,\ x, y \in T, x = y)$

Donde:

T = tipos de agente (conjuntos de roles o perfiles relacionados con ciertos objetivos, conocimientos o habilidades) Vamos a comentar algunas consideraciones sobre la definición 10 de SMA:

- Se basa en el hecho de la sociabilidad, es decir, la creación de ensambles de Ag. Esto confiere a los agentes la posibilidad de modularizar las soluciones, tal como se haría en los enfoques de programación convencionales, como la estructurada y la orientada a objetos, con la ventaja de que los elementos que conforman el sistema pueden ser agentes o grupos de agentes de tamaño diverso según se requiera.

- Su interacción se da en *un mismo ambiente*. Si los agentes operan en ambientes distintos, podrían ser sólo agentes interactuantes, pero no partes de un mismo SMA. Recordemos que un ambiente es la pregunta para la que los agentes se plantean como la solución; luego, al estar en distintos ambientes, responden a preguntas distintas, aunque esto no obsta para que puedan interactuar, como ya se dijo.

- Tienen objetivos en común, alcanzables de modo cooperativo o competitivo. De esta forma, se pueden crear estrategias de solución a un problema que explote el actuar en oposición de los agentes en pro de la consecución de un objetivo, o también complementando sus potencialidades cooperativamente.

- Hay *comunicación* entre ellos (generalmente directa y explícita). Como en todo sistema modular, los componentes intercambian información para monitorear sus estados, la consecución de sus fines y participar de los estímulos e intercambios con el ambiente. Es explícita cuando se dan intercambios *ex profeso*, pero también pueden ser de naturaleza indirecta

2. Interacciones y comunicación en un SMA

Cada agente dentro de un SMA realiza operaciones de comunicación, que pueden ser coordinación, completencia, planeación, cooperación y/o negociación, entre otras.

Sobre las interacciones entre los agentes de un SMA, que, como ya se ha dicho, son fundamentales para su operación, hay que considerar lo siguiente:

- Pueden ser *predefinidas* cuando son realizadas de acuerdo con protocolos de comunicación que fijan las condiciones y significados de la comunicación.

- Cuando no se pueden anticipar (en el caso de sistemas complejos): estas señales se tienen que resolver "al vuelo", es decir, dicho intercambio se debe procesar para asignarle un significado y proveer una respuesta conforme a estos estímulos sin un plan predefinido o un estándar a seguir.

- Pueden ser *directas* (mediante mensajes interdirigidos agente-agente, agente-sistema-agente) o *indirectas* (utilizando el ambiente).

2.1. Lenguajes de Comunicación entre Agentes

Aemás de las relaciones en un SMA es necesario comentar sobre las formas de comunicación interagente. Dichas comunicaciones pueden darse mediante protocolos definidos *ex profeso* implementados a través de Lenguajes de Comunicación entre Agentes LCA o ACL (por sus siglas en inglés: *Agent Communication Language*), algunos de los más famosos son KQML KQML y FIPAFIPA. En la tabla 5.1 se muestra una breve descripción de estos lenguajes [39, 8, 2].

LCA	Descripción
KQML	Del inglés *Knowledge Query and Manipulation Language*, Lenguaje de Consulta y Manipulación de Conocimiento, permite a los agentes compartir conocimiento "al vuelo", es decir en tiempo de ejecución. Utiliza como esquema de comunicación las llamadas *performativas*, es decir, instrucciones no evaluables como falsas o verdaderas descritas en la teoría de *actos del habla*.Los mensajes se estructuran en capas de contenido que incluyen las performativas, la identificación del emisor y receptor, y otros parámetros de gestión de mensajes. Algunos ejemplos de performativas son: *ask-if, ask-about, reply, broadcast*, y *tell*, entre otras.
FIPA	Fue creado por la Fundación para los Agentes Inteligentes o FIPA por sus siglas en inglés (*Foundation of Intelligent Physical Agent*); representa una mejora al anterior lenguaje al añadir nuevos parámetros, junto con la posibilidad de que el usuario pueda definir los propios. Algunos ejemplos de performativas son: *agree, disagree, confirm, cancel*, entre otras.
Otros lenguajes	Además de los métodos convencionales como $CORBA$, RMI, o RPC, se han utilizado como LCA aquellos basados en objetos, como el Protocolo Simple de Acceso a Objetos o SOAP (del inglés *Simple Object Access Protocol*); o en lenguajes de marcado de hipertexto o XML (del inglés *eXtensible Markup Language*).

Tabla 5.1: Lenguajes de Comunicación entre Agentes más relevantes en SMA.

2.2. Comunicaciones indirectas

Otras formas de comunicación pueden darse de forma indirecta, utilizando un medio o el ambiente mismo. A continuación se presentan dos formas: el esquema de pizarrón y la estigmergia.

Estructuras de pizarrón

En un primer caso, nos encontramos frente a un esquema de **pizarrón**, es decir, un medio centralizado para organizar unidades solucionadoras de problemas (como los agentes de un MAS), aunque hay que comentar que no pertenece estrictamente al modelo de éste, sino que más bien respondería a una necesidad de implementación, ya sea por la dificultad de comunicación o de almacenamiento (piénsese por ejemplo en sistemas embebidos).

Cabe señalar que estos esquemas pronto cayeron en desuso entre otras causas por las mayores capacidades de cómputo y el refinamiento de los modelos y arquitecturas de agente. Sin embargo, se mencionan aquí porque actualmente se han vuelto a presentar, si bien bajo esquemas más robustos, como el del cómputo ubicuo.

El pizarrón es una memoria de lecto-escritura global, por ejemplo una Base de Datos que sirve a un espacio de soluciones organizadas en una o más jerarquías de aplicaciones interdependientes, que la utilizan como repositorio y fuente de conocimiento (que pueden ser agentes). Cada una de estas aplicaciones utiliza un motor de inferencias particular, es decir, se realiza un razonamiento heterogéneo y no existe un control centralizado; el flujo de control es dinámico y no predefinido.

Se compone de tres niveles: aplicación (el programa para resolver el problema), estructura (el enfoque o especificación de las herramientas para resolver el problema) y el modelo (la descripción abstracta general del problema).

Algunas de las ventajas y desventajas de los esquemas de pizarrón son los siguientes:

- El comportamiento que soluciona el problema está disperso en el sistema como un todo.
- Las soluciones parciales escalan de forma incremental a una solución global.
- Es independiente de la tarea: clasificación, diseño, diagnóstico, reparación, etc.
- Admite unidades de solución heterogéneas y con niveles distintos de abstracción.
- Pueden presentarse cuellos de botella debido a la centralización de la Base de Datos.
- Requiere que los individuos lleven a cabo operaciones de integración de sistemas o utilicen un único lenguaje de acceso a datos y comunicación.

Estigmergia

El caso de la *Estigmergia* del inglés (*Stigmergy* y éste, a su vez del griego *stigma* (marca), y *ergon* producto, trabajo) [40] como elemento de comunicación en SMA se retoma del trabajo de Grassé [41] en entomología. Postula un mecanismo de comunicación basado en señales dispuestas en el ambiente, o que utilizan el ambiente como medio de dispersión, primeramente observado en insectos como las hormigas, pero que también se ha observado en ambientes humanos, sean o no ingenierados (como los sistemas de señalética), e incluso en la *web* se puede notar este mecanismo.

La estigmergia puede verse como un mecanismo por el cual la modificación del ambiente conduce a disparar o posibilitar ciertas acciones: un semáforo en rojo/verde; una señal de vuelta prohibida o el sentido de una calle; la marca de ocupado en un avatar de programa conversacional (*chat*), entre muchas otras. Puede verse como un puente entre el comportamiento individual y global: la acción de un agente (modificar parte del ambiente) puede influir en los comportamientos de otros agentes.

Algunas características relevantes son:

- Requiere o promueve interacción sin comunicación directa
- Las interacciones individuales o entre pares pueden conducir a un comportamiento global.
- Puede implementarse en formas muy básicas que no requieran de gran procesamiento, es decir, los agentes que la utilicen pueden ser también elementos muy simples.
- Puede sumar a un trabajo en marcha.
- Se puede instrumentar de diferentes maneras:
    - Cuantitativa: una marca perceptible o señal cuantificable.

  - Cualitativa: conjuntos discretos de tipos de estímulos: manipulación del material del ambiente, que podría no requererir una noción de agente, o una muy básica tipo agente reflejo.

- Los agentes pueden responder a los estímulos sin ser reprogramados, y aun cuando varios fallen, la tarea podría realizarse por el número de ellos que estén implicados (lo que se conoce como *degradación gentil* del sistema y *tolerancia a fallos*)
- Puede generar una alta dependencia de los parámetros del modelo del SMA, pues es difícil dividir un problema en partes implícitas (comportamientos individuales)
- Necesita un criterio para definir cuándo se termina el proceso, así como la imposibilidad de anticipar el comportamiento.

En la siguiente sección se abunda sobre los mecanismos de cooperación y coordinacion en SMA.

2.3. Cooperación y coordinación

Como ya se ha indicado, las relaciones entre agentes dentro de un SMA pueden ser de cooperación o coordinación, y ambas pueden alcanzarse por distintos medios, de forma directa o indirecta. Esto es por negociación y aún por competencia autointeresada.

En la Figura 5.2 se muestra una taxonomía de las interacciones posibles dentro de un SMA. Tanto cooperación como coordinación se utilizan para orga-

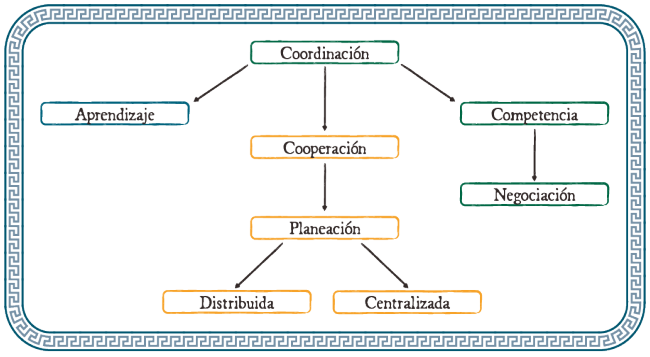


Figura 5.2: Taxonomía de formas de interacción en torno a la coordinación en un SMA

nizar a los agentes de un SMA. Para ello, en los SMA deben considerarse algunos aspectos:

- El número de agentes que se disponen para solucionar el problema en cuestión (aquello para lo que el SMA está diseñado o una parte específica de la solución esperada).
- La asignación de tareas y recursos a los agentes implicados.
- La coordinación/cooperación de actividades interagentes.

De esta forma, también hay que considerar que la estructura organizacional óptima depende del problema a resolver y las condiciones ambientales.

En cuanto a la **coordinación**, los agentes trabajan conjuntamente, o sea, se reparten tareas y comparten información entre sí para la consecución de la tarea general (el problema global), es decir, realizan una resolución distribuida de problemas.

Un ejemplo de coordinación lo podemos encontrar en un SMA que modele el comportamiento de un grupo de rescate urbano en caso de desastres. De este modo, la situación implica la conformación de grupos de salvamento y unidades de apoyo. Con esto se compone el siguiente escenario:

- Elementos implicados: unidades de rescate, base de operaciones (provee refugio, suplementos médicos, combustibles, entre otros), ambiente urbano (edificios, calles) y víctimas (movilidad reducida y existencia posiblemente comprometida).

- Habilidades básicas de los agentes: ubicación y desplazamiento en el área de desastre, detección de otros agentes y del ambiente, implementación de comportamientos específicos.

- Procedimiento de rescate: los agentes que conforman las unidades de salvamento operan en una área de desastre. Buscan víctimas que poner a salvo. Es decir, implementan un comportamiento o protocolo de rescate: remover escombros, limpiar calles, prover primeros auxilios, medicinas, extracciones físicas de personas atrapadas, entre otros.

Como se puede observar, la necesidad de coordinación surge al momento de que las unidades de salvamento implementen el procedimiento de rescate, por ejemplo, para una tarea básica: los encargados de remover escombros deben identificar cuáles son los lugares que deben ser despejados para permitir el paso de las unidades que llevan civiles a la base de operaciones. Para este caso, podría emplearse comunicación directa entre las unidades para saber cuál ruta debe utilizarse y si requiere ser despejada.

En el caso de la **cooperación**, se lleva a cabo un proceso de alineación o sincronización de las actividades de los agentes o grupos de ellos para trabajar juntos, comúnmente utilizando algoritmos específicos como las *intenciones conjuntas* (los agentes se orientan a la consecución de la misma tarea, quizá mediante acciones compartidas) o planeación parcial global (un plan general se realiza modularmente, estando a cargo de grupos o individuos).

Un ejemplo de esta interacción se puede retomar de lo expuesto en [42]; tomando un caso práctico, sería un asistente de correo electónico implementado como un SMA cuyo propósito fuera la realización de ciertas acciones en función del correo recibido en los buzones de los usuarios, imitando las acciones que éstos realizan con sus correos: enviar una respuesta predefinida, asignarle ciertas etiquetas para organizarlo, marcarlo como importante, o clasificarlo como *spam*, entre otros. En este sistema, los agentes podrían colaborar mediante el intercambio de información entre ellos (datos relevantes como direcciones electrónicas, patrones de texto, o etiquetas de correo electrónico; o incluso las propias reglas de acción). Dicha interacción podría establecerse de manera directa, vía un protocolo de mensajes interagente. De esta forma, los agentes aprenderían de los demás, beneficiando su propio rendimiento sin tener que inferir las reglas de

accion solamente a partir de los patrones observados en la conducta del usuario. El protocolo podría constar de intercambios disparados en caso de que un agente no sepa cómo resolver un caso, haciendo una petición de sugerencias a una colectividad de agentes registrados en una cierta cartera de agentes. Cada agente sugerente podría marcar su colaboración con un nivel de pertinencia estimado. Adicionalmente, la confiabilidad de las respuestas podría establecerse mediante mecanimos de reputación, donde las peticiones interagentes fueran evaluadas por el solicitante en aras de puntuar mejor aún la pertinencia de las sugerencias enviadas por sus pares.

También se ha comentado, en lo que respecta a la cooperación entre agentes, que no necesariamente tienen el mismo fin autointeresado; podrían cooperar o competir, pero también **negociar**. En esta forma de interacción se lleva a cabo un regateo entre los agentes dentro de un esquema de posibilidades y reglas, frecuentemente siguiendo ciertos protocolos para negociar tareas o recursos, máxime cuando éstos últimos son escasos o de difícil acceso.

En el caso de la **competencia**, ésta puede dar lugar a una cooperación como resultado global, sin que haya alineación explícita de actividades, es decir, un conjunto de agentes autointeresados puede conseguir sus propios objetivos y, al hacerlo, cooperar con otros simplemente porque determina que eso le ayuda a sus propios fines, no por la resolución del problema *per se*, sin embargo, ése es precisamente el resultado.

Consideraciones interesantes sobre cooperación, negociación y coordinación en SMA:

- La cooperación implica realizar *modelos* de otros *Ag* y de sus futuras interacciones. Es una forma de coordinación entre agentes no antagónicos.
- La negociación es coordinación entre *Ag* competitivos o auto-interesados. Es el proceso por el que dos o más *Ag* alcanzan una decisión conjunta (cada uno de los cuales conserva sus propios objetivos).

Ahora bien, hay una situación que es deseable en un SMA: **coherencia**, es decir, que un SMA se comporte como un todo unitario, pero sin que medie un control global explícito.

Con las interacciones anteriores, que eventualmente llegan a la resolución de una tarea, se puede emplear un mecanismo de *aprendizaje automático* para conseguir, por ejemplo, que los integrantes del SMA puedan coordinar sus acciones. Comúnmente, esta coordinación se puede alcanzar con enfoques de *aprendizaje por refuerzo* para evaluar la relevancia de sus acciones y organizarlas para proporcionar secuencias de operación apropiadas. Cabe señalar, una vez más que las interacciones que pueden aprenderse son de caracter variado, incluyendo la competencia y la negociación.

Finalmente, las interacciones entre agentes de un SMA tienen las características siguientes:

- La comunicación posibilita la coordinación de acciones y comportamientos.
- La coordinación sostiene la coherencia de un SMA (unidad de objetivo).
- La cooperación se refiere a la realización de alguna actividad en un ambiente compartido.

- El grado de cooperación permite:
  - Evitar malfuncionamientos (por comportamientos extraños).
  - Disminuir la contención de recursos y los abrazos mortales.
  - Incrementar las condiciones de seguridad.

2.4. Heterogeneidad en SMA

De lo que se ha dicho anteriormente, se destaca que un SMA puede estar compuesto por agentes de distinta índole y características. Esto es, contener agentes homogéneos o heterogéneos. La heterogeneidad u homogeneidad se sustenta en la variedad de reglas, propiedades, roles, capacidades que configuran un comportamiento por un lado más reducido o más robusto, pero también diferenciado y con más ricas y numerosas interacciones. Esto en conjunto confiere más posibilidades de reproducir un comportamiento global orientado a la solución de problemas. (similares en cuanto a sus capacidades y su diseño, que además de cumplir tareas diferenciadas y poseer capacidades distintas, se realiza bajo los principios de diversas arquitecturas). Por ejemplo, en la figura 5.1 se puede observar una representación de agentes que a primera vista podrían parecer homogéneos; pero, para efectos ilustrativos, los colores podrían actuar como diferenciadores de clases de agentes, es decir, como factor para denotar su heterogeneidad, que podría ser de objetivos (captar distintos tipos de alimento: círculos de otros colores, por ejemplo).

Ahora bien, algo en lo que se ha insistido es en las características de **homogeneidad** o **heterogeneidad** en un SMA, porque es una de las que permiten una mayor flexibilidad a estos sistemas para modelar la solución a diversos problemas y a su propia operación. De esta forma, un SMA puede conformarse por agentes diversos, entendiendo que la diversidad deviene de alguna de estas dimensiones:

- En términos de la similitud o diferencia de sus *arquitecturas*, puede haber agentes:
  - Reactivos: se basan en un modelo reflejo del mundo, comúnmente construido por reglas *si-entonces* para generar sus respuestas a los estímulos provenientes del ambiente.
  - Cognitivos: poseen un modelo deliberativo más robusto, comúnmente basado en lógica de primer orden, lógica difusa o modelos no simbólicos (Redes Neuronales Artificiales o Algoritmos Genéticos).
  - Híbridos: que integran características reactivo-cognitivas.
  - Y de todas sus subclases, además de otras arquitecturas.
- En cuanto a sus *habilidades* (todos saben hacer lo mismo o hay agentes con distintas habilidades).
- Por los *objetivos* o *propósito* final que persiguen: pese a tener metas distintas o incluso opuestas, podrían estar tras el mismo objetivo general (del SMA, o de los agentes individuales que lo conforman).

Por principio de cuentas, de los agentes reactivos revisados en los capítulos anteriores, hay que decir que, pese a su facilidad de implementación y efectividad, son insuficientes para representar los detalles de los individuos o de los conglomerados humanos.

De forma alternativa, se han propuesto arquitecturas que intentan estar más *ad hoc* con la representación de ciertos aspectos del comportamiento humano, conocidos como *agentes cognitivos*, cuya arquitectura ofrece mayor robustez que los reactivos, principalmente por sus modelos de deliberación. Por su nivel de complejidad, estos agentes se revisarán con mayor detalle en la segunda parte de esta obra.

2.5. SMA y Sistemas Complejos

La idoneidad de los SMA para el estudio de Sistemas Complejos se puede inferir a partir de diversas características que comparten y sobre todo tomando en cuenta que los SMA son en sí mismos sistemas que ostentan cierto un grado de complejidad, ya sea por el número de interacciones que sostienen sus elementos o por lo intrincadas que pueden resultar éstas al momento de estar operando.

- Son un método robusto y accesible para representar y analizar Sistemas Complejos o ciertos aspectos de ellos.
    - Sostienen relaciones y grados de complejidad diversos.
    - Sus interacciones toman en cuenta la temporalidad.
    - Los Ag y los SMA son en sí sistemas complejos.

Además, es conveniente hacer notar dos Nociones derivadas de la complejidad[1], que se presentan en los Sistemas Complejos en mayor o menor medida:

- Autoorganización: proceso que permite a los sistemas de software alterar dinámicamente su organización interna (estructura y funcionalidad) en tiempo de ejecución sin ningún mecanismo de control externo.
- Emergencia: proceso que permite a un sistema de software generar fenómenos emergentes (realización de funciones orientadas a la solución de problemas, de forma no pre diseñada, sino resultado de la dinámica del sistema).

En cuanto a lo que implica la heterogeneidad/homogeneidad en los SMA, con ella se pueden observar e implementar diferentes reglas, propiedades y capacidades, lo que deriva además en una ampliación de las posibilidades de comportamiento y, en consecuencia, mayores posibilidades de resolución de problemas vía un comportamiento global; además de un mayor número de interacciones, que conlleva a mejores condiciones para el surgimiento de la emergencia. Sólo que hay que notar que ésta se puede dar pese a que no haya reglas de comportamiento distintas, es decir, se puede presentar incluso en sistemas con reglas de comportamiento similares. En la tabla 5.2 puede verse una breve comparativa entre sistemas complejos y SMA.

[1] Conocidas en inglés como *Self-Organization* y *Emergence*.

Sistemas complejos de software	Sistemas Muli-Agente
▪ La complejidad forma una o varias jerarquías. Partiendo de estados simples o intermedios, el sistema evoluciona rápidamente hacia otros de mayor complejidad ▪ Las relaciones son dinámicas ▪ La versión primaria de los sistemas es definida de manera arbitraria por el diseñador (observador externo) ▪ Se suceden relaciones distinguibles en los subsistemas, tanto directamente entre ellos (más predecibles) como al interior de ellos (menos predecibles)	▪ Pueden concebirse fácilmente como una jerarquía ▪ Descomposición y abstracción: el problema es dividido en partes, resueltas por separado, lo que facilita el diseño ▪ Organización: los agentes pueden coalicionar, aislarse, cooperar, y agregarse a un sistema según sus necesidades, formando grupos o subseistemas que forman un todo permitiendo así la descripción de un sistema en términos de niveles macro-micro ▪ Se establecen algunas relaciones (dependencias e interacciones) entre los agentes en tiempo de diseño y ellos mismos pueden manejarlas dinámicamente. Asimismo, existen varios protocolos de comunicación entre agentes

Tabla 5.2: Características de los SMA que los hacen adecuados para el desarrollo y simulación de sistemas complejos.

2.6. Ambientes Multi-Agente

De igual manera que con los agentes individuales, en los SMA el ambiente conforma una parte indispensable para su operación y determinan cómo deben concebirse en el momento de diseño, pero sobre todo representan es escenario donde los agentes operarán para dal complimiento a sus propósitos comunes.

Recapacitemos en que el ambiente por excelencia es el *mundo real*, el cual es un entorno estilo SMA, es decir, no podemos ignorar a los demás agentes para lograr metas, pues si bien se pueden cumplir metas ya sea de forma individual o autointeresada, o bien adoptando una conducta competitiva que aproveche los recursos y oportunidades del ambiente; algunas de ellas pueden lograrse sólo con la cooperación de los otros.

Los Ambientes en entornos Multi-Agente poseen ciertas características que deben considerarse tanto para su diseño como para su análisis y operación:

- Proporcionan una infraestructura que especifica protocolos de comunicación e interacción.
- Son abiertos y no tienen un control centralizado.
- Contienen Ag autónomos y distribuidos (auto-interesados o cooperativos).

En la tabla 5.3 se puede ver una resumen de las principales características de los ambientes y sus implicaciones para los SMA.

Características	Implicaciones
Autonomía de diseño	Plataforma, protocolos de interacción, lenguaje, arquitectura interna
Infraestructura de comunicación	Memoria compartida (como el pizarrón, que cayó en desuso pero no está totalmente descartado), basado en mensajes (síncrono-asíncrono), conexiones variadas (punto a punto, multidifusión,...)
Protocolos de mensajería	KQML, HTTP / HTML, OLE, CORBA, ...
Servicios de mediación	Basados en ontologías, transacciones, otros
Servicios de seguridad	Autenticación, cifrado, marcas de tiempo
Soporte de operaciones	Almacenamiento y respaldos, redundancia, recuperación
Ambiente teleológico	¿Los agentes son parte de una mismo ensamble o sólo comparten el ambiente?

Tabla 5.3: Ambientes en SMA, sus características y sus implicaciones.

3. Modelación Basada en Agentes

Un modelo es una herramienta que permite aproximarse a un fenómeno en aras de su entendimiento y posible explicación. Los modelos se hacen tomando en cuenta ciertas características del fenómeno estudiado, es decir, se lleva a cabo una discriminación de la realidad. Para construir un modelo, comúnmente se emplea la *abstracción* para realizar dicho acercamiento, lo que implica dejar de lado ciertos aspectos, ya sea estructurales, composicionales, de escala, entre muchos otros.

Una manera de iniciarse en el modelado es realizar aproximaciones de aquello que se quiere entender, mejor dicho, de algunos aspectos de lo que se está indagando, mismas que pueden ser matemáticas o computacionales. A partir de la modelación se pueden reproducir los fenómenos estudiados mediante una *simulación*. Por tanto, la modelación y la simulación, en su vertiente computacional, puede realizarse utilizando el enfoque de agentes, lo que da lugar a la Modelación o Simulación Basada en Agentes.

3.1. Simulación Basada en Agentes

La Modelación o Simulación Basada en Agentes MBA) es una técnica computacional orientada a la representación y explicación generativa de sistemas complejos [43]. Está fuertemente ligada a la idea del surgimiento *emergente* de comportamientos complejos durante el curso de la simulación. Mediante ésta se pueden observar *comportamientos globales* a partir de acciones simples de sus componentes individuales. En otras palabras, los agentes (y sus comportamientos) trascienden la mera composición de partes individuales en un todo (el SMA). Además, estos comportamientos emergentes globales se presentan en ausencia de un control centralizado y en un ambiente en constante cambio. En la siguiente sección se enfatizará sobre la relación entre SMA y sistemas complejos.

La MBA permite la representación directa de los elementos del sistema así como de las interacciones que se realizan interagente y con el ambiente donde están situados. Se toma en cuenta la dimensión temporal, lo que posibilita observar los fenómenos emergentes que ahí se desarrollan. Por todo lo anterior, la

técnica en cuestión se perfila como la más idónea para la *representación de sistemas sociales*, con mayores ventajas que la que ofrecen los modelos puramente estadísticos, o bien, de aquellos basados en la teoría de juegos [43].

Para las Ciencias Sociales y las Humanidades, representa una herramienta de investigación empírica interdisciplinaria, cuya potencia clave radica en que posibilita representar con distintos niveles de detalle tanto individuos como sus interrelaciones y los ambientes en los que el fenómeno se desarrolla, lo que es útil en la confrontación de teorías sociales [44].

Como cualquier modelo computacional, las MBA requieren datos, los cuales pueden ser obtenidos de fuentes reales, como los provenientes de las propias investigaciones sociales: demográficos, sociológicos, económicos, entre otros. Es en este sentido (haciendo uso de una metáfora): análoga tanto al *microscopio*, en cuanto permite analizar los detalles más representativos de entidades sociales o individuos, y sus relaciones; como al *telescopio*, en cuanto permite observar dichas características en el conglomerado completo donde se ubican, incluso dentro de una cierta temporalidad.

En resumen, la MBA:

- Es un enfoque computacional en el cual se construyen simulaciones de procesos sociales y naturales, en un intento de entenderlos mejor.
- Permite que se simulen las consecuencias de la aplicación de políticas públicas, o fenómenos biológicos y sociales, como el impacto y, la transmisión de enfermedades entre la población; además de distintos fenómenos físico-químicos, entre muchos otros.

3.2. MBA: ¿por qué un programa de computadora?

Se dijo en un inicio que la simulación en general, y en particualar la social, podría realizarse mediante aproximaciones matemáticas y otras técnicas pertenecientes a las ciencias sociales [43].

El primer enfoque es lo que se conoce como *deductivo*, cuyas herramientas son los modelos matemáticos, ecuaciones y postulados de la economía clásica; sus asbtracciones son simples y elegantes y su poder predictivo consiste en que entre los varios estados que se pueden alcanzar (equilibrios) como resultado de la aplicación de las ecuaciones se pueden deducir consecuencias para mejorar la comprensión de los fenómenos.

El segundo enfoque, el *inductivo*, es más apegado a la tradición sociológica: realiza generalizaciones a partir de observaciones que pueden variar en su extensión e intensidad dependiendo de una variedad de factores (alcance o posibilidad del investigador de acercarse al fenómeo estudiado, disponibilidad de datos y observaciones, entre otros). Los hallazgos que pueden alcanzarse son de carácter cualitativo, describen los hechos mediante la caracterización y elaboración de categorías generales.

Ahora bien, un tercer enfoque es el *computacional*, que es claro que podría emplearse de manera muy conveniente en la simulación social, puesto que permitiría ahorrar muchos recursos y reproducir eventos y fenómenos, que por su naturaleza, serían imposibles, tanto material como moralmente (piénsese tan sólo en los conflictos armados). Su idea central es construir un programa que modele el comportamiento de un fenómeno social; esto es, realizar un modelo

computacional de éste. Y su propósito es entender el fenómeno/sistema y quizá poder prever o predecir su comportamiento o parte de él. Una simulación de este tipo permite iniciar con supuestos explícitos y, en lugar de demostrar teoremas, se producen datos analizables de forma inductiva, que posteriormente hacen plausible la generalización de sus consecuencias. Los datos conforman reglas preespecificadas, no provienen de observaciones directas, pues el propósito es modelar lo que se está estudiando. De esta forma, se puede aplicar la deducción también, para encontrar las consecuencias de los supuestos iniciales.

De forma sintética, de los tipos de modelación podría decirse lo siguiente [43, 44]:

- Las descripciones verbales son flexibles, pero imprecisas.
- Las descripciones matemáticas son rigurosas, pero es difícil incluir heterogeneidad en ellas, amén de la no-linealidad y la tratabilidad.
- Los modelos computacionales son rigurosos, precisos, facilitan representar heterogeneidad e interacciones, y permiten la experimentación, así como combinar tanto inducción como deducción.

3.3. Elementos de una MBA

Un caso especial de la simulación computacional es la que se basa en agentes para realizar su modelado, es decir la MBA, que posee los componentes que se enlistan a continuación:

- Agentes
- Ambiente
- Comportamiento
- Interacciones
- Tiempo

En lo sucesivo, se amplían las características de cada uno de ellos.

Agentes

¿Cuál es la razón del uso de agentes?, en primer lugar, hay que considerar a los agentes como "unidades de comportamiento", es decir, tomadores individuales de decisiones: son autónomos, heterogéneos, aprenden, tienen memoria, interactúan y modifican el ambiente, todas ellas características analizadas en el capítulo 3.

A través de los agentes se pueden representar:

- Gente: votantes, consumidores, pacientes, contribuyentes, ejidatarios y terratenientes...
- Empresas y organizaciones diversas (como las Organizaciones No Gubernamentales, sindicatos, etc.).
- Países, estados, alcaldías, municipios, demarcaciones territoriales, entre otros.

- Conceptos e instituciones: prácticas sociales, usos y costumbres, leyes, etc.
- Objetos del ambiente: animales, células, elementos geográficos y urbanos, por citar sólo algunos.

Ambiente

Al igual que en los SMA, los ambientes son cruciales para representar un fenómeno, incluso en capítulos previos se indicó que el ambiente es el problema para el cual el agente es la solución.

Pero resaltemos algunos hechos sobre el ambiente en una MBA:

- Es donde están situados los agentes y puede ser físico o virtual, pero en cualquier caso posee, en mayor o menor medida, ciertos aspectos de la realidad, o bien *es* la realidad de los agentes: el sistema de archivos de un servidor, o el planeta Marte, por dar dos ejemplos.
- Típicamente, cuando son virtuales, pueden tatarse de espacios bidimensionales o incluso en tercera dimensión, y son representados comúnmente como cribas de objetos con ciertos atributos.
- Denotan espacios geográficos o físicos como países, regiones, edificios, complejos habitacionales, etc.; pero también pueden ser espacios conceptuales: opiniones, redes de colaboración, cadenas tróficas, redes sociodigitales, etc.
- También pueden no tener un significado específico (cuya definición o significación puede no estar incluida como parte de la simulación).

Comportamiento

Ahora bien, ¿quién ejecuta el "comportamiento" en una MBA? En primer lugar, diríase que los agentes, sin embargo, hay varios niveles de comportamiento, por lo que también se pueden considerar distintos tipos de aproximaciones a los sistemas modelados; debido a esto, se hablaría del macrosistema, es decir, el todo completo (la MBA); el mesosistema, es decir una aproximación más a nivel de componentes grupales o individuales (agentes-ambiente); y el microsistema, que va más dentro aún de los componentes individuales y diferenciables del sistema (elementos constitutivos de los agentes u objetos del ambiente).

De esta forma, tenemos lo siguiente:

- Agentes: son los encargados de percibir el ambiente, toman decisiones y lo modifican con sus acciones: ejecutan secuencias de acciones conforme sus objetivos de diseño, sus capacidades y sus recursos disponibles.
  - Ejemplos: comprar acciones, dirigirse a un lugar, tomar vacunas, cuidar sus cultivos y casas.
- Ambiente: se compone de una estructura y una variedad de objetos que sostienen una dinámica propia; a través de ellos responde a las acciones de los agentes y a su misma "naturaleza".
  - Ejemplos: el fuego creciente o decreciente en un incendio, un comando del sistema operativo que es invocado, el cambio de posición de un objeto al ser impelido por un agente.

Interacciones

Recordemos que las interacciones son un punto crucial en los SMA, lo que no exenta a la MBA, que es un subconjunto de los primeros. Entonces, ¿quiénes y para qué interactúan? Observemos lo siguiente:

- Típicamente, las acciones detonan otras acciones y éstas, a su vez, se ven afectadas por ellas, es decir, no están aisladas.
- Ejemplo: la tirada de un oponente en el ajedrez, la colaboración de otro agente al realizar una tarea.
- En una MBA pueden existir estos dos tipos de interacciones:
  - Indirectas: el ambiente pudo cambiar por la acción de otros.
  - Directas: las decisiones/comportamiento del agente modifican el ambiente.

Por lo tanto, se puede afirmar de manera categórica que si una MBA no tiene interacciones, quizá no sea el enfoque apropiado para modelar el problema/sistema bajo estudio.

Tiempo

Finalmente, ¿la temporalidad en que transcurre un fenómeno importa? Uno de los aspectos más importantes para considerar el uso de una MBA y, al mismo tiempo, una de sus características más relevantes es que permite trabajar la temporalidad del fenómeno, ya sea a través de una representación fiel de los eventos o mediante una abstracción relevante.

De esta forma, hay que atender ciertas consideraciones sobre el tiempo en una MBA:

- En una MBA, el tiempo transcurre, es una **simulación** de un fenómeno, no un sistema estático.
  - Típicamente, este tiempo es medido en lapsos/períodos discretos (como los instantes), pasos o *tics* de reloj.
  - Cada instante, *tic* o paso de tiempo representa un ciclo de comportamiento de objetos del ambiente o agentes, o bien, de toma de decisiones para estos últimos.
- A su vez, este tiempo medido sirve para que se desenvuelva o norme la toma de decisiones, esto es para medir y ordenar el comportamiento esperado y/o observado.
- Ejemplos: los agricultores deciden cuándo sembrar (no lo hacen diario ni cada hora), un cuidadano vota ¿en cada periodo electoral?

De igual manera, si la temporalidad interesa en el fenómeno estudiado, el enfoque MBA resulta muy útil.

Complejidad

Adicionalmente, se puede observar que una MBA, al ser de suyo un SMA reproduce ciertos aspectos de la complejidad del fenómeno estudiado, y es en sí mismo un sistema complejo. De esta forma:

- Una MBA incluye la **complejidad** del mundo real en el ámbito del fenómeno que se representa.
- En los aspectos que se representan, la heterogeneidad y las interacciones son clave para su desarrollo y desenvolvimiento.
- Las interacciones que se dan en su seno incluyen la realimentación y dos fenómenos importantes:
  - Emergencia de productos o procesos, que son más que la suma de las partes.
  - Autoorganización, denotada por cambios en el comportamiento o la estructura del sistema, con el fin de adaptarse al ambiente (a sus fluctuaciones) y seguir operando.

¿Y eso qué provoca? Es decir, las consecuencias de que se presenten tanto la emergencia como la autoorganización son principalmente:

- No hay punto de predicción, cuando mucho estimación gruesa o relativa.
- La emergencia se observa sólo durante la operación, no puede ser explicada a partir de la descripción de sus componentes.
- Emergencia y autoorganización pueden ocurrir juntas, pero también de forma separada.

4. Desarrolo de MBA

En esta sección se dará una breve introducción al desarrollo de una MBA, utilizando el lenguaje de programación y ambiente de desarrollo NetLogo.

Razones para una MBA: ¿cuándo y por qué?

Existen algunas razones para desarrollar una MBA:

- Interés en modelar interacciones y realimentación entre actores, y entre los actores y su medio.
- Si la heterogeneidad de los actores es importante en el sistema.
- Si nos interesa la dinámica espacial del sistema.
- Si la historia (acciones pasadas) son importantes para el desarrollo de los acontecimientos presentes/futuros del sistema.
- Si los actores tienen comportamientos cambiantes o se adaptan a lo largo del tiempo.

- Si queremos utilizar un enfoque intuitivo y flexible para un modelado participativo.

¿Qué tenemos hasta el momento? Básicamente una idea de qué son las MBA y cuándo son útiles:

- Cuando se necesita: entender y estimar el comportamiento de un fenómeno.
- Cuando el fenómeno a considerar posee una heterogeneidad en sus elementos.
- Se dan variadas o intensivas interacciones entre ellos.

Ahora pasemos a entender cómo se pueden construir.

4.1. Implementación

¿Y cómo se realiza la simulación?, ¿con qué herramienta o lenguaje simulamos? Desde el punto de vista de la implementación computacional, hay varios enfoques:

- Diseñando, codificando, modelando...
  1. Desde cero: utilizar un lenguaje de programación de propósito general (C/C++, Python, Java, ...).
  2. En un ambiente especializado para MBA (hay gran diversidad).
- NetLogo (https://ccl.northwestern.edu/netlogo/):
  - Es un ambiente de desarrollo y modelado para SMA.
  - Autoría e implementación: Uri Wilensky/Center for Connected Learning and Computer-Based Modelling, Universidad de Northwestern (EE. UU.).

NetLogo es uno de los más populares: cuenta con acceso abierto, con una gran comunidad de usuarios, lo que garantiza soporte; es fácil de codificar y aprender. Además, es un ambiente orientado a lo gráfico. Su comunidad provee además de soporte una amplia variedad de ejemplos y modelos. Y es de destacar que tenga varias prestaciones: capacidades de interacción con herramientas estadísticas (como el lenguaje R), proporciona formas de almacenar las salidas de las simulaciones y de ciertas tareas tanto en texto como en imágenes. Y una facilidad interesante es que permite prediseñar experimentos y recabar información sobre ellos.

Para una introducción a los rudimentos del lenguaje, se pide al lector revisar el anexo B.

4.2. Ejemplo de una MBA en NetLogo

Para iniciarse en la implementación de una MBA se propone revisar un modelo ya establecido en el acervo de NetLogo, que además es muy relevante por la importancia que tuvo cuando se dio a conocer y la sencillez de sus componentes.

El ejemplo se denomina *Segregación*, y se inspira en un modelo propuesto por Thomas Schelling[2] sobre la preferencia de que personas de distinto color (se ubica en Estados Unidos, donde la principal oposición de lo que se conoce como "raza" socialmente hablando se da entre afrodescendientes ("gente de color") y anglosajones ("blancos"), se toleren como vecinos.

Este modelo permite observar la segregación en cualquier ciudad. Contiene dos tipos de agentes (representan dos grupos de personas): rojos y verdes. Al observar el modelo, la pregunta natural para el lector es ¿qué microdinámica produce las macrocaracterísticas observadas?

Abrir y correr el modelo:

1. Abrir el modelo en NetLogo, esto es, navegar por el menú: NetLogo- Archivo>Biblioteca de Modelos>CienciasSociales>Segregación, como se muestra en la figura 5.3.

2. Pulsar botón de configuración e inicio: *Setup*, *Go*, tal como se ve en la figura 5.4

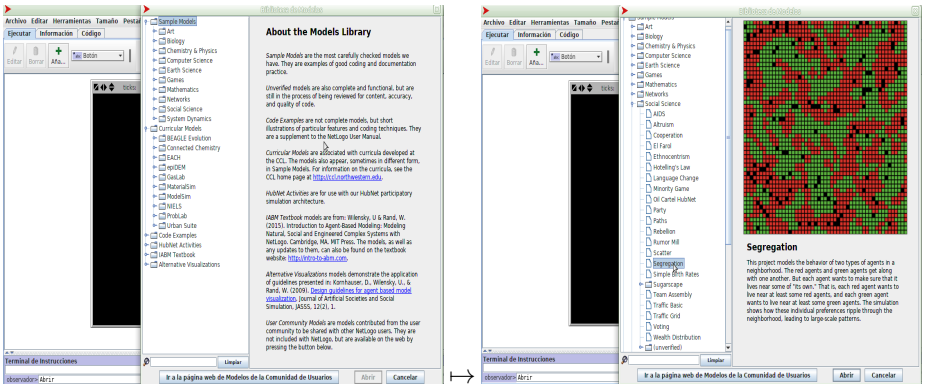


Figura 5.3: Abrir el modelo "Segregación" en NetLogo.

Una vez que el modelo se ha abierto, se puede verificar que al inicializarlo hay una mezcla de agentes rojos y verdes.

El propósito de la operación será observar cómo se agrupan estos agentes de acuerdo con las reglas de tolerancia a lo distinto y el azar con el que se dispone al inicio las vecindades entre ellos.

Este modelo tiene ciertas características dada su dinámica muy simple, es decir, su comportamiento:

- Su distribución inicial es aleatoria.
- Cada grupo tiene una cierta tolerancia/preferencia por la heterogeneidad de su vecindario.
- Si su tolerancia es superada por vecinos que no son de su agrado, se muda.

Ahora, ¡a experimentar! Veamos los resultados al variar la tolerancia ($\%similar-wanted$). Finalmente, pregúntese el lector ¿qué se puede aprender del modelo?

[2] https://es.wikipedia.org/wiki/Thomas_Schelling

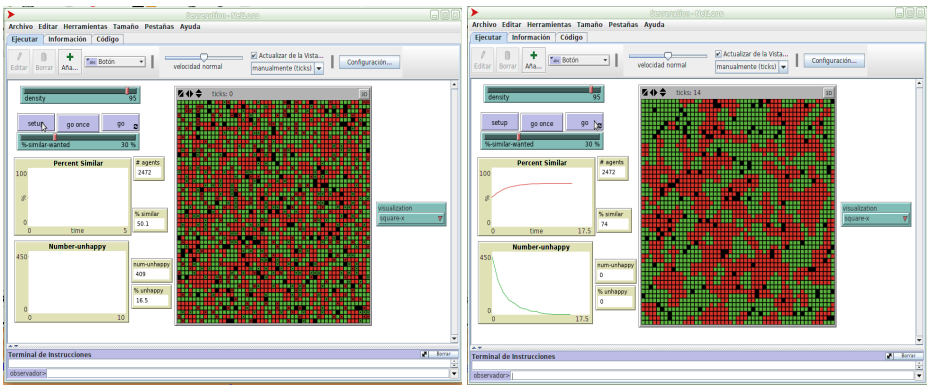


Figura 5.4: El modelo "Segregación" en operación cuando ya se puede ver la segregación de verdes y rojos.

5. Resumen

Sistemas Multi-Agente y Simulación Basada en Agentes

En este capítulo se describieron los SMA y la MBA

- Los SMA explotan la característica de intercomunicación y habilidad social de los agentes para formar conglomerados de ellos, comparten un mismo ambiente y un propósito común .
- Pueden estar formados por agentes homogéneos o heterogéneos.
- Se pueden unir en su operación mediante cooperación, ya sea a través de la colaboración o la competencia.
- Su intercomunicación puede ser predefinida, directa o indirecta mediante diversas técnicas. La estigmergia es una de las cuales es materia de investigación y ha mostrado ser prometedora.
- Los MBA son en sí sistemas complejos donde se dan los fenómenos asociados a la complejidad: autoorganización y emergencia.
- La MBA es un área prometedora por su capacidad de representar distintos conglomerados naturales y sociales, la escala de los fenómenos, su dinámica y temporalidad.
- NetLogo es una plataforma de SMA, específico para realizar MBA, que incluye tanto un entorno de desarrollo y experimentación como un lenguaje de programación de agentes reactivos.

6. Actividades de aprendizaje

6.1. Ejercicio de implementación

Es un ejercicio guiado de caracter introductorio, enfocado en que el alumno se adentre en la confección del programa de una simulación en el entorno de NetLogo.

Objetivo

La actividad consiste en cargar un modelo en NetLogo, y hacerle modificaciones directamente en el entorno.

Descripción

Este modelo tiene dos tipos de agentes diferenciados por color (rojos y verdes). Los agentes se mueven en el ambiente y al interactuar entre sí cambian de color de rojo a verde y viceversa.

Desarrollo

Se deben realizar los siguientes pasos:

- Reproducir los controles mostrados en las imágenes de este ejercicio.
- Observar que el modelo a reproducir tiene una ventana de graficación para mostrar cómo varía la población de agentes rojos y verdes, como en la Figura 5.7.
- ¿Qué debe hacerse?
    1. Reproducir el modelo, para lo cual debe abrirse un archivo y comenzar desde cero tanto interfaces como programación.
    2. Crear controles para observar el desarrollo de la simulación.
    3. Implementar las reglas de interacción de los agentes:
        - Deben moverse aleatoriamente.
        - Si están junto a una mayoría de otro color, adoptar éste.
- ¿Cómo debe hacerse?: poner las siguientes entradas y controles.
    1. Selección de la forma del agente, usar *persona* o *default* (ver Figura 5.8)
    2. Dial para determinar la población inicial (rojos y verdes).
    3. Interruptor para indicar si el agente deja huella o no (ver igura 5.9).
    4. Añadir las salidas necesarias: gráficas de población de agentes rojos y verdes, y dos cajas de salida para ver los números de agentes por color en el momento actual (ver figura 5.10).
    5. Añadir una etiqueta para explicar brevemente el modelo.
    6. En cuanto al código del comportamiento, se debe hacer lo siguiente:

        a) Crear la función `creaAg` para crear los agentes.

b) Crear la función `inicio` que establezca el ambiente y cree agentes.

c) Añadir un botón `Inicio` y asociarlo a la función `inicio`.

d) Crear la función `correr` para iniciar el comportamiento de los agentes.

e) Añadir un botón `Dar un paso` y asociarlo a `correr`.

f) Implementar la regla de interacción mediante funciones.

En la figura 5.5 se puede ver el modelo operando.

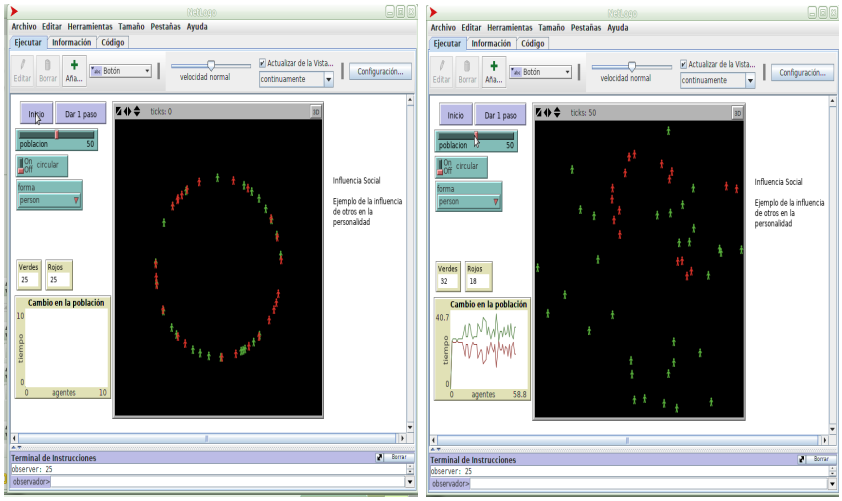


Figura 5.5: El modelo Influencia Social (versión didáctica mínima).

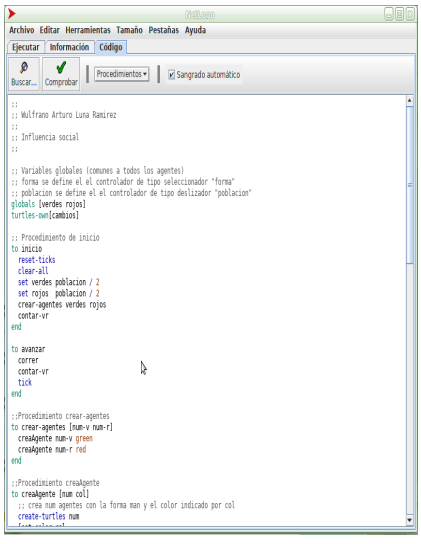


Figura 5.6: Código del modelo Influencia Social.

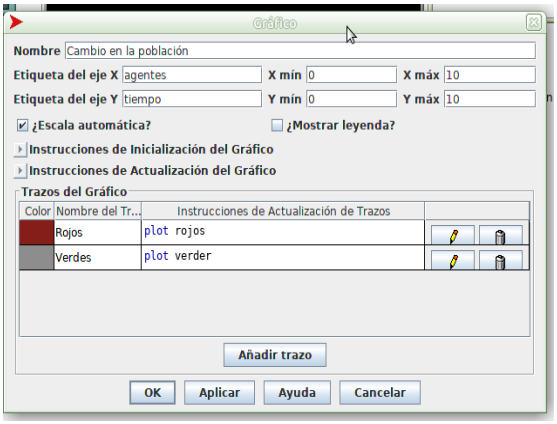


Figura 5.7: Trazos de número de agentes rojos y verdes.

Figura 5.8: Selección de figuras para los agentes.

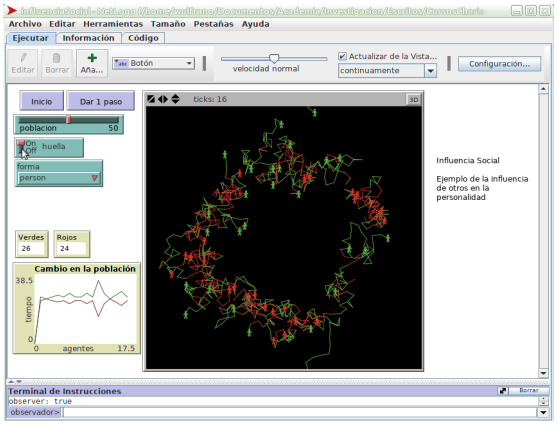


Figura 5.9: Huella de los agentes en el ambiente del modelo Influencia Social.

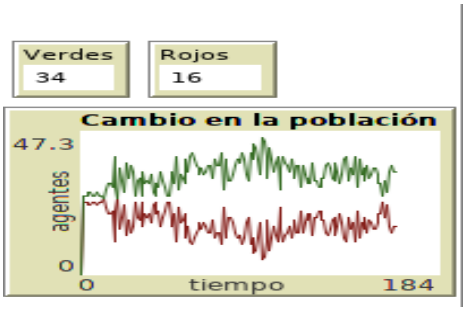


Figura 5.10: Graficación de la salida del número de agentes.

Finalmente, consideremos sólo algunos otros aspectos que pueden observarse en el modelo, como la visualización 3D, la ejecución continua, es decir, que la función principal del programa se ejecute de manera indefinida, y la velocidad de movimiento de los agentes. Todos ellos controlables y modificables por el entorno de NetLogo (ver la figura 5.11)

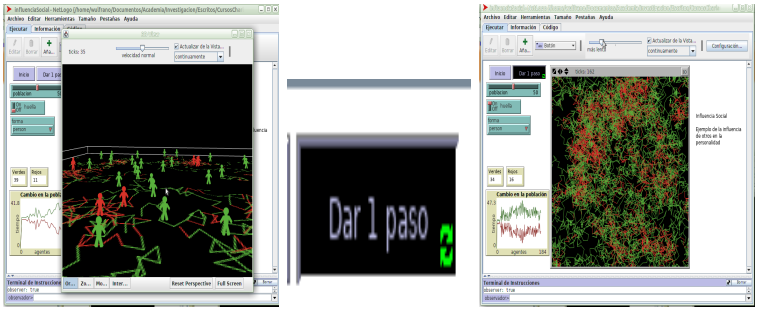


Figura 5.11: Otras pretaciones que ofrece NetLogo: vista 3D, reproducción continua y velocidad de los agentes

Para efectos de verificación, se remite al lector al punto 2 del anexo B, donde está el listado del código resultante (únicamente de las funciones): B.1. Por lo que el lector podría:

- Cargar el archivo del modelo *InfluenciaSocial*, del anexo B [3]: *influenciaSocial.nlogo*, tal como se ve en la figura 5.5.
- Revisar el código del programa, usando la solapa *Código* (ver figura 5.6)
- Considerar que el código mostrado no incluye lo concerniente a la interfaz gráfica, sino sólo la funcionalidad.

[3] Disponible en línea: `pendiente`

Capítulo 6

El horizonte humano de la IA y los SMA

Objetivos

- Conocer las consecuencias y alternativas a la noción valorativa de la IA.
- Identificar las características de la crítica decolonial al desarrollo de la IA.
- Conocer la alternativas para llevar la IA a una escala humana.

> *Según la historiadora de la inteligencia artificial, Pamela McCorduck, los más tempranos ejemplos de la urgencia humana por construir seres artificiales se encuentran en La Ilíada.*
> *Pero McCorduck no conoce el Popol Vuh y, por tanto, ignora que antes que La Ilíada fuese escrita, en lo másprofundo del corazón de la materia, Tepeu, el Creador, y Gucumatz, el Formador, ordenaban, componían y recomponían átomos y moléculas; ensayaban formas, ecuaciones, conexiones y acoplamientos varios.*
>
> *El mundo en un grano de maíz, Manuel Martínez Morales*

Durante los capítulos precedentes se dio una breve introducción a la IA y a los SMA con énfasis en la MBA. Se han aportado hasta aquí deficiones teóricas y detalles técnicos que llevarían al lector a una comprensión mínima del campo de los agentes y los multiagentes, y a su programación en al menos dos lenguajes de programación: Python y NetLogo.

En el presente capítulo se esbozan perspectivas sobre el campo de la IA y los SMA a la luz de consideraciones de contexto que desembocan en análisis ético-morales que no son tangenciales al desarrollo científico-tecnológico por las razones que a continuación se exponen, y que por el contrario, pueden decantar el camino a seguir para la evolución de los sistemas inteligentes tal como ahora los conocemos.

1. La fascinación por lo *artificial*

Como se señaló desde el primer capítulo, una constante en la historia de la humanidad es lo que podríamos denominar como su fascinación por lo artificial. Esto se ha manifestado en las antiguas máquinas chinas, entre los griegos que dejaron para la posteridad evidencia de sus máquinas mecánicas, y actualmente lo podemos ver en diversos espacios de la vida, como los dispositivos embebidos que permean gran parte de los objetos y espacios que nos rodean.

Un ejemplo de esta fascinación pretérita y contemporánea se encuentra los llamados edificios inteligentes; recordemos que en Alí Baba y los cuarenta ladrones ya se enunciaba un mecanismo operado por reconocimiento de voz que al "ábrete sésamo" daría acceso a su interior. Otro ejemplo para evidenciar la aludida fascinación: recordemos cómo son recibidos entre el público los autómatas europeos hechos con mecanismos de relojería, antecedentes de los modernos robots y programas de IA, que constatan la universalidad de esta inquietud humana.

Pese a lo que se ha señalado anteriormente, es necesario identificar que el desarrollo de la IA, en tanto tecnología, pero también ciencia, se ciñe a condiciones que van más allá de lo puramente técnico. En primer lugar, habría que destacar que si bien el ímpetu por la generación de entidades artificiales, como ya se dijo, parece estar presente en todas las culturas, ni el desarrollo de la IA ni sus frutos se reconocen en igual medida.

1.1. La IA, ¿global?

Algunos de los textos clásicos que abordan la historia de la IA, como el de Pamela McCorduck *Machines who think* [45], o la historia intelectual de la IA dibujada por John Haugeland *Inteligencia Artificial*[46] o el homónimo recuento realizado por Margaret A. Boden [47], por citar algunos de los más conocidos, plantean un panorama muy esclarecedor de las ideas que sostienen el desarrollo de laIAA. El problema es que en ellos persiste una visión completamente eurocéntrica que nos borra casi completamente de este recuento, lo que hace necesario retomar dicha historia intelectual incorporando lo que en otras tradiciones científicas y esfuerzos tecnológicos, además de las visiones filosóficas y sociales de diversas sociedades (dentro de las cuales estamos insertos en México y Latinoamérica).

La situación planteada anteriormente, además de incorporar los tintes propios de nuestras comunidades, recuerda que aun el pensamiento eurocéntrico ha obtenido, muchas veces sin citar los orígenes, ya sea por por omisión o por ignoracia [48], intercambios científicos y culturales que lo ha enriquecido. Tal estado de cosas apunta a que ha hecho falta un esfuerzo continuado de describir los aportes que nuestras civilizaciones pueden hacer al desarrollo de la IA, además de las contribuciones actuales para confeccionar la disciplina actual que conocemos.

La IA en México, una nota breve

Como un beve recuento de lo que se ha hecho en nuestro país, se puede señalar que nuestros esfuerzos en IA orbitan alrededor de unos pocas instituciones que sostienen ciertos intereses de investigación y de docencia.

Desde los albores del cómputo en nuestro país, que comenzó a finales de la década de 1950, tanto la investigación como la docencia se han ejercido mayoritariamente en instituciones públicas dentro de las que pueden destacarse por sus cursos y programas de docencia, y su volumen de desarrollos y productos de investigación: la Universidad Nacional Autónoma de México, el Instituto Politécnico Nacional, la Benemérita Universidad Autónoma de Puebla, el Centro de Investigación y de Estudios Avanzados, y el Instituto Nacional de Astrofísica, Óptica y Electrónica. Mención especial merece la Universidad Veracruzana y el Laboratorio Nacional de Informática Avanzada, instituciones que crearon uno de los primeros posgrados dedicados exclusivamente a la IA, vigente hasta la fecha.

En lo que se refiere a las líneas de investigación que se cultivan en las instituciones antes mencionadas se puede observar que los temas que favorecen abarcan tanto cuestiones teóricas como aplicaciones tecnológicas aplicadas en temas que podrían aglomerarse (no de manera limitativa) en: Procesamiento de Lenguaje Natural, Visión Computacional, Robótica, Reconocimiento de Patrones, Optimización y Cómputo Bioinspirado.

Sin menoscabo de los esfuerzos señalados, es claro que esta ciencia y sus tecnologías derivadas aún deben desarrollarse y las instituciones además de multiplicarse, deben fortalecerse. Afortunadamente, cada vez es más frecuente que se creen opciones de docencia y grupos de investigación en nuestro país cuyo objetivo es el cultivo y desarrollo de la IA.

2. Por una IA a escala humana

En esta sección se plantean algunas directrices en el desarrollo de la IA. Para dar inicio, se propone revisar la pretendida neutralidad de las ciencias todas, pero en particular de la IA, a la luz de problemas no sólo tecnológicos o ético-morales, sino también sociales que se hacen presentes en la actualidad, además de abordar las preguntas de qué rasgos socioculturales está reflejando la IA y las opciones para incluir las particularidades de nuestras poblaciones. Todo esto está motivado por una notoria ausencia de representación en los desarrollos y productos tecnológicos y científicos del área que será señalada en lo sucesivo. Se parte, entonces, de la observación de esta carencia y las posibilidades de su superación.

2.1. Problemas del enfoque de desarrollo actual

Como ya se ha dicho anteriormente, la IA es un reflejo del pensamiento y la inteligencia humana, a eso aspira y en ella toma fuerza e inspiración para hacer muchos de sus planteamientos, y como tal, cada cultura plantea una manera de pensar distinta y añade sus matices culturales. Aunado a lo anterior, la mayoría de los adelantos en el área se originan en los países dominantes política, económica y militarmente. Además, dichos países cuentan con las instituciones académicas de mayor demanda y actividad en el área, y muchos de nuestros investigadores se han formado en ellas.

Hasta este punto, podría afirmarse válidamente que el fenómeno descrito anteriormente configura un intercambio cultural que nos es único de la IA ni de nuestro tiempo, se ha dado en muchas áres y disciplinas a lo largo de la historia.

Sin embargo, cuando dichos investigadores regresan a sus lugares de origen (en el caso de México y Latinoamérica hay una gran porción de ellos que no vuelven a sus países), llevan consigo una impronta cultural y modelo de hacer investigación y docencia que, consciente o inconscientemente, reproducen en las instituciones en las que se afincan. Si bien parece que no hay estudios respecto de los impactos que tal fenómeno suscita, convendría pensar un poco en sus implicaciones, no para sostener un endogamia que podría incluso ser nociva, sino para identificar, al menos en el caso de la IA, cómo influye en su desarrollo y aplicaciones.

Así, podríamos pensar en que los conocimientos y desarrollos no sólo deben ser adoptados (aún si son adaptados a nuestras realidades y problemas) sino en cómo desarrollar una IA que refleje nuestra forma de pensar y nuestros rasgos culturales (en particular por la gran riqueza cultural de nuestras comunidades originarias). Sobre todo, deberíamos pensar en conseguir una mejor y mayor adecuación del desarrollo de la IA a nuestra realidad.

Por estas dos razones, es necesario preguntarse hasta qué punto los desarrollos de la IA reflejan el pensamiento hegemónico, no ya sólo de las sociedades donde se desarrolla, sino de las grandes empresas trans y supranacionales que la desarrollan y explotan los beneficios de sus productos comerciales.

2.2. La ideología tecnológica

Con el objetivo de entender cómo llevar a cabo un cambio de rumbo en el desarrollo de la IA, se debe empezar por analizar cómo es actualmente el desarrollo

científico y tecnológico. Para ello debe aboradarse un tema que pocas veces se toca en la disciplina: falsa neutralidad ideológico-política de la ciencia y la tecnología (CyT), y por ende de la IA, pues ésta es una ideología en sí misma que dificulta la valoración de otras formas y aproximaciones científico-tecnológicas, las de la "periferia", donde nos ubicamos. En esta tarea, el pensamiento filosófico de nuestros maestros iberoamericanos es de gran ayuda como se verá enseguida.

De acuerdo con el filósofo hispanomexicano Adolfo Sánchez Vázquez [49], tal ideología es la que sostienen muchos de los devotos de la falsa neutralidad ideológico-política de la CyT. El autor refiere que la **Ideología Tecnológica** (IT) asume ciertas posturas, a saber:

- Supone la autonomía del desarrollo tecnológico: asume que la tecnología sigue su curso de forma independiente respecto del acontecer de las sociedades en las que se desarrolla.

- Ve a la tecnología como un fetiche: supone que la tecnología por sí misma basta para solucionar cualquier problema, ya sea social o de índole diversa.

- Le adjudica a la tecnología el concepto de dominio: considera que la dominación de un grupo social, sea una comunidad, un país o los Estados-Nación, es sólo una consecuencia del desarrollo tecnológico.

- Pretende que existe una desideologización de la tecnología: cree que el desarrollo tecnológico es apolítico y libre de cualquier ideología, sea política, religiosa, social o económica.

El problema, nos dice el autor, es que estas posturas, amén de ser falaces, encubren relaciones de dominio: ni la tecnología controla al hombre (algo que es particularmente relevante hoy en día con el auge de la IAG), ni está desligada de ciertos fines (muchos de ellos económicos y de control) ni de sus ideologías (incluyendo las neoreligiones).

Como ya se indicó, es de crucial importancia el caso del dominio, pues ahora se hace más que patente que el poder económico termina por convertirse en poder dominante. Poco a poco ha sido evidente cómo las grandes empresas tecnológicas se han convertido en entidades ya no sólo trans sino supranacionales con una acumulación de capital abrumadora que, es necesario advertirlo, no proviene sólo de su dominio técnico, sino que su origen está sustentado en grandes inversiones. Dicha base económica en algunos casos supera en tamaño al de países enteros.

De esta manera, estos grandes actores conforman grupos de presión económica y política, y terminan por convertirse en la firmas dominantes, mismas que dejan sentir su influencia más allá de su sector. Eso les posibilita cambiar y aún imponer las reglas comerciales y económicas de distintos sectores y empresas (competidoras o no). Esto les confiere también la posibilidad de intevenir directa o indirectamente en las sociedades y países a tal grado que ha habido variadas voces en torno a que de no regularse dicho actuar se ponen en riesgo los órdenes democráticos liberales [50], algo de lo que no se escapan ni los mismos países donde se incubaron y radican las grandes tecnológicas.

Ante este escenario, vale la pena observar algunas alternativas para supera la IT y sus efectos:

- Superar la *neutralidad valorativa de la ciencia* [51], que consiste en asumir como posible la supuesta limpieza ideológica de la ciencia, y pretender que ésta no tiene influencia en pro de los intereses de grupos empresariales, políticos o económicos.
- Considerar la CyT como un sistema de acciones intencionales [51, 52]: esto es, asumir que el sujeto y sus intereses importan, tienen una intención y un propósito.
- Analizar lo que Sánchez Vázquez [49] identificó como *racionalismo tecnológico*, que ostenta una doble arista indivisible:
  - La racionalidad tecnológica (teórica): propia de la cuestión meramente operativa o instrumental, de funcionamiento técnico, ése sí fuera del alcance de las ideologías.
  - La racionalidad de los fines (práctica): referida a la cuestión de *para qué se desarrolla algo*, influíble totalmente por los fines y objetivos de los actores implicados.
- Orientarse al *cambio de fines* en pro del autodesarrollo individual y social. Esto es lo que Sánchez Vázquez señaló como **una CyT escala humana**.

Entonces, en el nivel de la racionalidad práctica, ha de revisarse la legitimidad de los fines de la CyT (y por ende, de la IA): ¿a quién sirve?, ¿cuáles son los fines que la motivan?, ¿cuáles son los tintes ideológicos sobre los que subyace el enfoque y metodología?

3. Una IA decolonial

Tras lo anterior, para conseguir superar la IT que atañe también a la IA, es necesario echar mano de ciertos preceptos de la teoría decolonial, para ir en aras de una autodefinición en el desarrollo de la IA en nuestras sociedades y países, con el objetivo de alcanzar una IA decolonial, y con ello posibilitar **una IA a escala humana**.

3.1. Un nuevo esquema de desarrollo

La **ia** debe adoptar las teorías del pensamiento crítico y de la decolonialidad, para superar buena parte de los problemas ya señalados, como han señalado recientemente autores como Mohamed y colegas [53], pero en un sentido integral, es decir, considerando la **ia** como un sistema de acciones intencionales que considere además de la legitimidad de los fines, el cuestionamiento y mejora de su racionalidad tecnológica y su fundamento científico, es decir, de sus base epistémicos y teórico-experimentales.

En este sentido, la esfera epistémica de la IA pasa por una discusión más integral y actual sobre el concepto de racionalidad. Es menester entonces retomar a los pensadores iberoamericanos que se han ocupado de la decolonialidad, del pensamiento crítico y otros temas relacionados con la autoafirmación intelectual y social de los pueblos y naciones. Esto arrojaría luz sobre la necesidad acuciante para la IA de, por un lado, retomar y criticar la historia del pensamiento y las ideas de la *modernidad* como único horizonte cultural, y aportaría

opciones también para superar la presunción de una exclusividad y supremacía del razonamiento lógico-matemático de corte aristotélico (y sus sucesores), con Kant, Hobbes, Descartes y sus continuadores intelectuales como bases inamovibles e inmejorables. Permitiría también eliminar o minimizar la exclusión de lo sensible y el desprecio por lo diferente.

Como ejemplo de la necesidad de repensar la racionalidad, hay que recordar al filósofo Enrique Dussel [54], quien citó a su vez a Fabien Eboussi Boulaga:

> El *yo pienso, luego soy* es el asesinato del yo danzo de Boulaga Eboussi... No he nacido para pensar, pienso para vivir [...] No *nacimos para argumentar*, *argumentamos para vivir*. Cuando danzo, no demuestro que existo, como la piedra, ¡no!, demuestro que vivo.

Es particularmente curioso que la postura anterior tenga su reflejo en controversia entre la IA Clásica y la Nueva IA o IA Situada. La IA Clásica, como se indicó en el Capítulo 1 se fundamenta en la *hipótesis simbólica*, que iguala la manipulación de símbolos con el acto/proceso de pensar, y por extensión, al ser considerado el cómputo una manipulación de símbolos, se asume que *computar sea igual que pensar*. Por otro lado, IA Situada sostiene que para desarrollar entidades inteligentes sería posible únicamente si se adopta el ambiente real (es decir, el mundo) como el escenario que moldee el comportamiento y la cognición de los agentes. Deja en segundo término la modelación basada en abstracciones y algunos enfoques los elimina totalmente, lo mismo con cualquier forma de modelado explícito [11]. Es decir, la IA Situada cuestiona la noción de racionalidad vigente en buena parte de los enfoques y concepciones de la IA, cuando menos para el desarrollo de la robótica.

Otro elemento a cuestionar, que es herencia directa de la ideología política del liberalismo es la consideración del individuo como actor central. Así, para la IA la inteligencia es individual. La ciencia ficción ha explotado bien esta idea, por ejemplo con el famoso *exterminador robótico* que se asume uno, solo, autosuficiente e invencible. Esto supone el desarrollo, por ejemplo, de una agencia artificial y una intencionalidad centrada en el individuo. De esta forma, la representación de estados mentales en los agentes artificiales sólo puede ser de carácter individual. Algo que puede y debe cuestionarse ampliamente.

Adicionalmente, en la esfera político-ideológica, los temas de la IA generan mensajes que pretenden la toma hegemónica del discurso, pero que de analizarse responden a la lógica ya señalada de la supeditación de la IA a la mercadotecnia:

- Nuevas retóricas y discursos hegemónicos: el prestigio que da el utilizar tecnologías, el *ser* inteligente... artificial.
- La falaz "dictadura" del algoritmo: el código es un reflejo del programador, de su jefe, de su gerente (CEO, en el argot tecnológico)... y de sus financiadores. Al dictum de Ada Lovelace habría que añadirle estos actores y sus intereses.
    - El algoritmo (de no estar pensado así) no incorpora el contexto social, geopolítico, económico, etc., y si lo hace, es sólo parcialmente (implicando únicamente lo relativo a aquello que le es importante).
- Las falsas expectativas y dilemas (como el de la singularidad) que configuran el preludio de otro invierno de la IA, pues sostienen situaciones que

rayan técnica y socialmente en lo inverosímil pero con un afán sensacionalista.

3.2. Alternativas decoloniales

Frente al contexto ya descrito, se pueden avisorar algunas rutas posibles para recorrer desde nuestros horizontes para acercarse a una IA de corte decolonial. Enseguida se analizan brevemente las dos caras de la racionalidad tecnológica referida a la IA: la teórica y la práctia. Se revisan de manera enunciativa sin aspirar a ser limitativos, pues sus posibilidades teóricas y sus cuestiones prácticas son muy amplias.

Desde la racionalidad teórica de la IA

Desde la propia racionalidad tecnológica de la IA pueden explorarse las potencialidades de algunas de sus subáreas, como las que se mencionan a continuación [25, 23, 3, 43].

- Cómputo de enjambres y distribuido: que ponen de manifiesto no la individualidad, sino la colectividad, tanto en términos de la concepción de la solución de problemas, como de su ejecución, complementaria y en cierto punto contraria a la centralización del cómputo y la toma de decisiones.
- Sistemas Multi-Agente y Simulación Basada en Agentes: que aborda ahora aunque de forma incipiente nuevos enfoques de la intencionalidad compartida, sistemas de coordinación y resolución colectiva de problemas, además de la propia representación de problemas particulares de índole colectivo y situado (como los problemas sociales).

En el segundo caso, se insiste en que los agentes artificiales, los SMA y la SBA pueden y deben incorporar nociones que trasciendan las que les son habituales (muchas de ellas desarrolladas a lo largo de esta obra) y operativizarlas, todo lo cual redundará en beneficios en el conocimiento, es decir, en la esfera de su racionalidad práctica propiamente dicha. Algunos ejemplos son los que se listan a continuación:

- Mecanismos de coordinación inspirados en la inclusión social y ecológica.
  - ¿Cómo operacionalizar conceptos de avanzada como el *mandar obedeciendo* o *el poder obedencial*? (consistentes no en un poder centralizado o personificado, sino distribuido y colectivo).
  - ¿Cómo incluir las nociones de pueblos originarios como el del *a'iel snopel* (sentir-pensar-escuchar) del pensamiento tsotsil [55] que apunta hacia una redefición de la inteligencia, el pensamiento y la racionalidad.
- Incorporar la preocupación ético-moral, valores y elementos de regulación social para normar el desarrollo de la IA y evaluar la conveniencia de sus desarrollos tecnológicos
- Desarrollar agentes inteligentes y SMA como una mediación digital, primeramente al servicio de los usuarios, no de las corporaciones.

Desde la racionalidad práctica de la IA

En lo que respecta a la racionalidad de los fines, se debería incentivar la apropiación de los desarrollos tecnológicos de la IA, además de enfocar su aplicación para apoyar procesos civiles, comunitarios y personales, tales como la gestión gubernamental, los derechos humanos, el mantenimiento de la privacidad física y digital, la seguridad humana, la soberanía alimentaria, la autodeterminación personal y de los pueblos, la preservación de la herencia cultural, entre muchos otros.

Se tiene que poner atención en ¿quién decide cuándo un problema debe ser solucionado?: la industria, los pares académicos, los editores tecnocientíficos, los gobiernos, las comunidades (que muchas veces financian los desarrollos con sus impuestos o colaboración indirecta o no remunerada, expropiaciones de tierra o bienes digitales, campañas públicas de información o publicidad, entre otros. Derivada de la anterior, ¿qué investigación o desarrollo tecnológico tiene prioridad o importancia?, ¿para quién?, y ¿cómo se sanciona ésta?

Aunado a lo anterior, desde la racionalidad de los fines, la IA decolonial debería revisar un antecedente inmediato: los fundamentos del software libre. Ellos ayudarían a establecer guías propias de desarrollo y evaluación, como el derecho a saber la manera en que los sistemas funcionan, y la capacidad de intervenirlo. Lo anterior podría pensarse también como un derecho ciudadano o una garantía individual.

Evitar el uso militar de la IA

Para cerrar este punto, se menciona que un aspecto crucial para el desarrollo de la IA es que deben incluirse consideraciones ético-morales que desincentiven su uso militar (particularmente el no regulado), por ejemplo, para prohibir el desarrollo y la proliferación de armas autónomas [56, 57, 58, 59], y sólo circunscribirlo al apoyo logístico [60, 61, 62]. El fundamento tras esta postura es el siguiente:

> Ninguna máquina es susceptible de responsabilidad moral, por lo que no pueden decidir sobre la vida de las personas, ni mucho menos sobre la muerte.

Además de contrarrestar en cierta forma la proliferación de las armas autónomas, esto pasa por asumir plenamente la convicción de considerar a la IA como un sistema de acciones intencionales, y no entenderla sólo como un entramado que opera bajo la racionalidad tecnológica que le es propia.

Entonces, se debe promover desarrollo y usos de la IA que oriente su *evaluación externa* incluyendo:

- Las diferencias culturales, geográficas, económicas, religiosas; ideológicas y tecnológicas [63, 64].
- Incluir una perspectiva de género.
- Promover el respeto irrestricto a los DD HH, derechos ambientales y de pueblos originarios.

Esto pasa por promover, en lugar de armas autónomas, el desarrollo de Agentes Artificiales y SMA como Artefactos Vitales Autónomos.

Como se ha ilustrado, tanto técnica como prácticamente existe la posibilidad de implementar postulados diferentes a los que rigen la investigación y desarrollo actuales de la IA.

3.3. Apropiación social de la IA y los SMA

Además de lo que se ha expuesto, es muy importante la promoción del desarrollo y apropiación social de la IA, cuyo desarrollo no debe estar sólo al alcance de la gran industria, sino que debe orientarse al servicio y uso de grupos y comunidades, personas interesadas, sin que se les considere sólo usuarios finales o consumidores, sino usuarios capaces de intervención en las tecnologías, de producir nuevas versiones y esquemas de prueba.

Una nueva *Drosophila* de la IA

Una avenida aún por transitar es que la IA se enfoque en la discusión y atención a problemas sociales, muchas veces tildados de locales, con un interés acotado. Sin embargo, hay que superar esta visión, pues resolver un problema local en muchos casos puede trascender a la resolución de problemas globales. En otras palabras, si un problema realmente lo es, es falso que se circunscriba sólo al ámbito local.

Otro argumento a favor de tomar los problemas sociales como un ariete en el desarrollo de la racionalidad instrumental y práctica de la IA y los SMA es la obtención de los siguientes beneficios:

- Trascender el problema de la escala: no caer en modelos funcionales, en problemas acotados de baja escala, sino tratar un problema en su dimensión real, si bien a través de un modelo o aproximaciones sucesivas.
- Enfrentarse a problemas complejos: los problemas sociales, son de suyo multifactoriales, dinámicos, jerárquicos y con varios componentes, en los que el factor humano complejiza los fenómenos.
- Apoyar la hipótesis situada: derivado del uso de problemas reales, el escenario de acción de los sistemas es el mundo real y su complejidad dentro de un contexto específico.
- Promover el desarrollo de la IA en comunidades en torno a problemas de interés compartido: apropiación de la IA no sólo como consumo, sino como herramienta, metodología, enfoque de trabajo, que puede ser intervenida y modificada según las necesidades encontradas.
- Pueden ser una fuente de financiamiento y desencadenar otros tipos de relaciones entre cuerpos de investigación-sociedad: es claro que un factor de financiamiento es la masificación de productos y servicios, y que vale la pena que las propias comunidades se hagan cargo cada vez más de ciertos aspectos de su desarrollo material e intelectual.

De esta forma, las posibilidades de pensar la IA y desarrollarla desde posturas no hegemónicas requieren de ciertas acciones:

Incluir la teoría crítica y decolonial en una IA pensada como un sistema de acciones intencionales. Analizar no sólo los fines o usos (sus productos), sino

su propia base teórica (su racionalidad científico-tecnológica). Promover la literacidad y su apropiación social: sustentabilidad (madre Tierra), cuestiones ético-morales, diversidad de género, organización social, entre otros. Modificar la *Drosophila* de la IA: transitar del ajedrez, del Go, de la mercadotecnia, a la resolución de problemas sociales. Retomar conocimientos y saberes de la periferia.

Finalmente, se hace énfasis en que debe cuestionarse la visión mercantilista actual de la IA, que se aleja de su pretensión de ser una ciencia que plantea el conocimiento de la inteligencia natural (primordialmente humana), así como las implicaciones que tiene que la ciencia y tecnología digitales, incluyendo a la IA, desde su producción, aplicación y uso generalizado, tengan otra visión además del horizonte de la plusvalía. La propuesta sería transitar de una IA orientada sólo hacia los fines de cierto modelo de capitalismo, a una que tenga como horizonte la humanidad.

4. Consideraciones finales

4.1. Los SMA frente a la IAG

Dado el auge presente de la IAG en variados aspectos del acontecer tecnológico, los Agentes Artificiales y los SMA no están exentos de la influencia que ésta ejerce.

A este respecto, recientemente se ha postulado el enfoque de los SMA como una manera muy prometedora para potenciar los alcances de la IAG. Esto consiste brevemente en que sean los SMA los encargados de gestionar los intercambios entre los bots de IAG y las peticiones de usuario [65]. Es decir, pasar de tener al usuario intercambiando consultas (*prompts*) con los *chatbots*, a que sean los agentes quienes gestionen las preguntas y respuestas, y verifiquen la corrección o validez de éstas.

En el primer caso, el usuario es quien interactúa de manera directa con las herramientas de IAG, analiza y se hace responsable de sus resultados [1], tal como se ve en la figura 6.1. En el segundo caso, como se ilustra en la figura 6.2,

Figura 6.1: Esquema de trabajo con la IAG actual: sin agentes.

[1] Recuérdese revisar el esquema de trabajo propuesto en el apéndice C para estas interacciones.

se introduce a los agentes para gestionar las salidas, y solicitar mejoras a los resultados obtenidos con las herramientas de IAG. Lo importante aquí es que el agente puede hacerse cargo vía sus habilidades y capacidades de cómputo de forma más eficiente y, en ciertos casos, precisa de los intercambios con dichas herramientas. A partir de estas dos versiones, sería interesante indicar que al

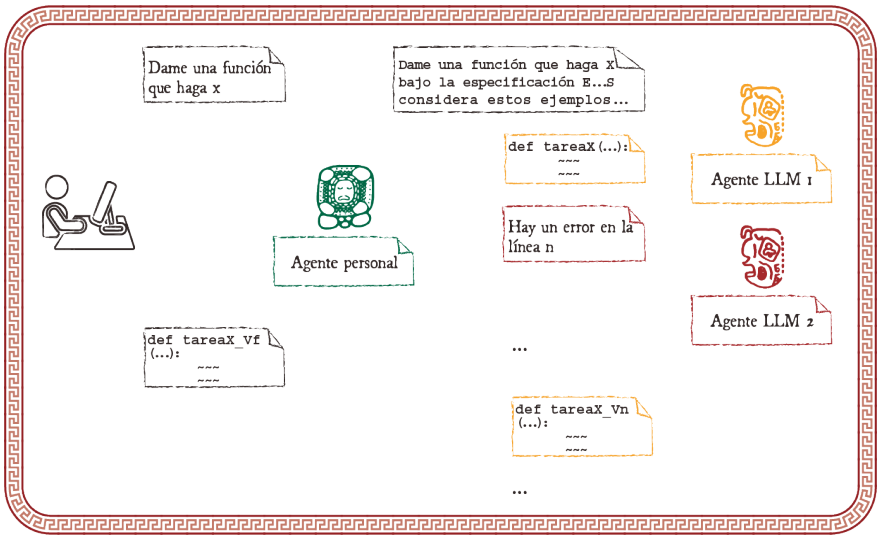


Figura 6.2: Esquema de trabajo con la IAG emergente: con agentes.

retomar los postulados de la sección anterior, los SMA pueden actuar como una mediación digital que obre en pro del interés de los usuarios a los que representa y sea éste quien gestione la interacción digital, incluyendo, por supuesto, la IAG, pero haciéndose cargo también de otros ciberintercambios, que velen así por la privacidad, derechos digitales y ciberseguridad. Este intercambio se puede ver en la figura 6.3.

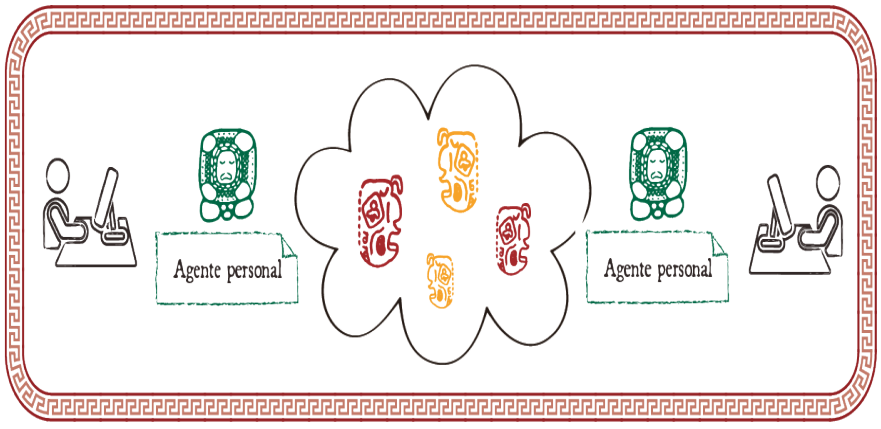


Figura 6.3: Esquema de trabajo con la propuesta de utilizar a los SMA como mediadores digitales.

4.2. Perspecivas de los SMA

Como ya se comentó, el área de los SMA ha cobrado nueva relevancia sobre todo por el enfoque que éstos proporcionan, por un lado la idea de que sea el mismo software el que solucione ciertos detalles de interacción digital y el usuario intervenga sólo en ciertas condiciones de importancia y complejidad, que no puedan dejarse en el ámbito de los sistemas de información.

De esta forma, a modo de cierre, se postulan alguna consideraciones sobre el presente y futuro de los Agentes Artificiales y los SMA.

- Son más que un programa de cómputo (aun si éste es distribuido): como se indicó en el Capítulo 1, los agentes y los SMA representan un paradigma con un potencial que aún no se ha explotado ni desarrollado del todo, pero que sigue siendo muy prometedor.

- Ayudan a modelar e indagar sobre la naturaleza humana, individual y colectiva: los modelos que se representan e implementan en los agentes, aún siendo muy limitados y orientados a aspectos y características computacionales, son una herramienta que permite imbuir a los sistemas de información de ciertos rasgos del comportamiento humano, individual y en grupo, como la toma de decisiones en situaciones restrigindas y de escasez de recursos (tiempo, materiales, espacios, opciones), o el comportamiento frente a y dentro de un grupo con intereses determinados (coincidentes o no).

- Son una propuesta de corte unificador en la IA: representan la opción de incorporar una diversidad de teorías, enfoques y metodologías de todas las áreas de la IA, tanto en su vesión de software (softbots), como en su cristalización en sistemas mecatrónicos (robots), ya que pueden insertarse en una diversidad de tareas e incluir en el proceso ramas tan desarrolladas como la Visión Artificial y el Procesamiento de Lenguaje Natural, en las formas de conseguir métodos automatizados de deliberación (que incluye Planeación y Aprendizaje Automático), entre otras.

- Son un enfoque potenciador de la IAG y las Tecnologías Inmersivas: aún está en ciernes la integración del enfoque de agentes en muchas de las tecnologías que se han desarrollado en la reciente década, como la IAG y las tecnologías de Realidad Virtual y Aumentada, pero que es muy factible que den lugar a una miríada de aplicaciones en diversos sectores y dominios.

- Representan oportunidades de desarrollo multidiscplinario, particularmente para áreas como la Ingeniería de Sofware y la Informática, y las Humanidades y Ciencias Sociales, sin menoscabo de las discplinas socioadministrativas y de negocio. Estas interacciones son necesarias y mutuamente beneficiosas, ya que hasta el momento sólo se tienen algunos intentos, muchos de ellos muy fructíferos, de incluir el conocimiento social y humanístico en los SMA, además de ser necesaria la creación de nuevos esquemas de negocio, que obren en pro de la humanidad toda y no sólo de ciertos sectores. En este sentido, la MBA es una de las principales áreas para echar a andar experimentos y generar nuevo conocimiento, por lo que se estima que los SMA sean una área que entre en auge próximamente.

5. Resumen

Una IA a escala humana

Pensar la IA a una escala humana requiere:

- Desarrollarla desde posturas no hegemónicas y promover la literacidad y su apropiación social.
- Superar la neutralidad valorativa de la IA (es un sistema de acciones intencionales).
- Analizar no sólo los fines o usos (sus productos), sino su propia base teórica y epistemológica (su racionalidad científico-tecnológica).
- Retomar conocimientos y saberes de la periferia e incluir la teoría crítica y decolonial.
- Complementar las *Drosophilae* de la IA: de los juegos a los problemas sociales.
- Los Agentes Artificiales y SMA configuran una posibilidad de nuevas aplicaciones e investigaciones en pro de una IA a escala humana, y para potenciar otros desarrollos como la IAG.

6. Actividades de aprendizaje

6.1. Reforzamiento cognitivo

Lleve a cabo las siguientes tareas para reforzar lo aprendido

- Realice un mapa conceptual con los conceptos clave del capítulo
- Redacte un breve ensayo indicando los aspectos de su entorno inmediato (lugar de residencia, escuela, trabajo, familia, comunidad) donde se puedan aplicar herramientas y tecnologías de IA, y particularmente, de Agentes y Sistemas Multi-Agente.
- Enliste las preocupaciones ético-morales que a usted le atañen en los desarrollo de IA y SMA

6.2. Preguntas

Conteste a estas cuestiones utilizando el criterio y palabras propias, reforzado por lo visto durante el presente capítulo. No hay un tipo de respuesta completamente correcta o incorrecta, pero hay que considerar que una buena respuesta es la que da lugar para el contraste y la argumentación, no sólo para las opiniones.

- ¿Cómo se explicaría la posición de la concepción de la ciencia y la tecnología como sistemas de acciones intencionales?
- ¿Cómo explicaría usted, con sus propias palabras, la posición de la concepción de neutralidad valorativa de la ciencia?
- ¿Cómo avisora el futuro e impacto de la IA a nivel internacional, a nivel nacional y a nivel personal?

Apéndice A

Representación en IA

En este anexo se muestran de manera muy breve dos modelos para representar problemas. Se trata de información de contextualización cuyo propósito se aleja mucho de ser exhaustiva, aún explicativa, más bien lo que se quiere es impulsar al lector a referirse a los elementos de la bibliografía aquí citada para ahondar más en las explicaciones. Quizá el lector más avanzado simplemente prescinda de esta contextualización, pero se deja al resto de lectores como una referencia. Tómese entonces como un material introductorio en complemento a lo que en el texto se utiliza.

1. Lógica

La ciencia del razonamiento, como se le ha denominado [1, 8], es uno de los formalismos más usados para modelar el conocimiento, pues es una manera precisa, fiable o modular de hacerlo [2, 8, 66]. En las postrimerías de la IA, fue uno de los paradigmas más favorecidos y aún persiste como una parte importante dentro del área. La Lógica ha sido estudiada desde los tiempos clásicos de la tradición cultural de Occidente, principalmente desde un enfoque filosófico que posteriormente se ha venido matematizando (formalizado).

La Lógica ha pasado por un proceso de depuración y maduración que le brinda robustez y la hacen una candidata aceptable como forma de representar el conocimiento y como un mecanismo de comprobación de hipótesis. Gracias a esto, se han podido implementar procesos de razonamiento y aprendizaje automático, entre muchas otras aplicaciones.

Revisemos los siguientes conceptos y definiciones para darnos una idea de lo que es la Lógica: una primera intuición de lo que es, nos dice que la Lógica es el estudio de los métodos y principios usados al distinguir entre los argumentos correctos (buenos) y los incorrectos (malos), un medio de clasificación de argumentos [1, 8]. Si atendemos a lo que dice Xirau, la Lógica (del griego *lógos*, razón) es la parte de la filosofía cuyo objetivo es el razonamiento recto y dirigido a encontrar la verdad y evitar el error [1].

Sin embargo, el proceso de matematización indicado anteriormente demuestra finalmente que la Lógica es más cercana a los métodos de las matemáticas que a la filosofía –sin que esto implique una ruptura–, adoptando el nombre de Lógica Matemática y supone un estudio del razonamiento humano con métodos

algebraicos, manteniendo como elementos cruciales el cálculo y la demostración exacta [8]. Ahora bien, toma el nombre de Lógica Simbólica del hecho de trabajar a partir de un lenguaje artificial y formal [8, 66], sin relación –directa– con la realidad, pues le conciernen (para hacer la clasificación antes mencionada) únicamente la forma de los razonamientos, no su contenido ni la manera en que fueron logrados, es decir, no el proceso de razonamiento en sí, sino, se insiste, su forma [1].

Esta manera de caracterizar a la Lógica, nos lleva a considerarla desde dos perspectivas:

- Es un instrumento orgánico para apreciar o evaluar la corrección del razonamiento.
- Es un objeto de investigación, en tanto que sus principios y métodos son materia de estudio sistemático. ¿Cómo y por qué es que puede ser un instrumento orgánico?, ¿bajo qué leyes y mecanismos?

A la Lógica le compete la corrección del proceso de razonamiento, una vez que se ha completado. Su pregunta por excelencia es ¿La conclusión alcanzada se sigue de las premisas usadas o supuestas? Si las premisas son verdaderas, es decir, dan fundamento a la conclusión, entonces se puede afirmar la veracidad de ésta, y el razonamiento es considerado *correcto*. A continuación se mencionan algunos conceptos relevantes de la lógica [1].

- Inferencia: es el proceso de afirmar un proposición sobre la base de otra u otras aceptadas como punto de partida.
- Proposición: toma valores de Falso o Verdadero. En el sentido de enunciado, son expresadas por las formulaciones lingüísticas declarativas. Una oración declarativa puede contener varias proposiciones y la variación de las oraciones (debida al contexto) pude afirmar distintas proposiciones.
- Argumento: es parte de la inferencia. Grupo de proposiciones o enunciados dentro de los cuales existe una que se sigue de los demás, que son su fundamento (le dan la verdad).
- Conclusión (de un argumento): es la proposición afirmada con base en las demás proposiciones del mismo argumento.
- Premisas (de un argumento): proposiciones que se afirman como fundamento o razones para la aceptación de la conclusión.

En la tabla A.1 se ejemplifica un argumento, sus partes y la manera en que pueden representarse simbólicamente, es decir, mediante variables sentenciales[1].

Hay que señalar que la premisa y la conclusión son relativos al argumento en que se empleen (dependen de la circunstancia, del contexto): una misma proposición puede ser premisa o conclusión en diferentes argumentos. Esto se ejemplifica en la tabla A.2.

[1] Este uso particular de los símbolos no es estándar, es una representación arbitraria para fines de los ejemplos de las tablas A.1, A.2 y A.3.

Proposiciones	Variables sentenciales
Todos los hombres son mortales	$h \rightarrow m$
Sócrates es un hombre	$s \equiv h$
Por lo tanto, Sócrates es mortal	$s \rightarrow h$

Tabla A.1: Argumento, premisas y conclusión expresadas como enunciados y como símbolos, esto es, con el uso de variables sentenciales. Adaptado de [1].

Proposiciones	Variables sentenciales
Todos los hombres son mortales	$h \rightarrow m$
Sócrates es un hombre	$s \rightarrow h$
Por lo tanto, Sócrates es mortal	$s \rightarrow h$
Todos los animales son mortales	$a \rightarrow m$
Todos los hombres son animales	$h \equiv a$
Luego, todos los hombres son mortales	$h \rightarrow m$

Tabla A.2: Argumentos donde una *misma proposición* es premisa o conclusión, de acuerdo con el uso en cada una de ellas. Adaptado de [1].

A continuación se menciona un aspecto básico de los argumentos, donde se muestra el cometido de la Lógica y el beneficio de su uso para representar el conocimiento:

Verdad vs. validez: se dice que verdaderas son las proposiciones o enunciados, y como tales pueden caracterizar a las oraciones declarativas que los formulan. Válidos son los argumentos (pueden ser caracterizados como) si sus premisas y conclusiones están relacionadas de manera que es imposible que las primeras sean verdaderas si la conclusión no lo es, se dice entonces: la conclusión se sigue de sus premisas. Es necesario señalar, a este respecto, que un argumento puede tener proposiciones verdaderas o falsas y ser válido. En la tabla A.3 se muestran un ejemplo de esto. De ahí que existan las posibilidades siguientes [1]:

- La validez de un argumento no garantiza la verdad de su conclusión.
- Empero, la falsedad de su conclusión indica que:
  - O el argumento es inválido o por lo menos una premisa es falsa.

Por lo tanto, para que un argumento tenga una conclusión verdadera se requiere que:

- El argumento sea válido (tarea del lógico y del lógico deductivo en particular).
- Todas sus premisas deben ser verdaderas (tarea de la ciencia en general).

Es por ello que se considera la validez (o invalidez) como una **característica puramente formal**. De esta manera, dos argumentos cualesquiera que guardan la misma forma o son válidos ambos o son inválidos los dos, independientemente de sus diferencias de contenido. En la tabla A.3 se muestran dos argumentos, el primero tiene premisas verdaderas, en tanto que el segundo las tiene exclusivamente falsas, pero ambos son válidos.

Proposiciones	Variables sentenciales
1	
Todos los murciélagos son mamíferos	$u \to m$
Todos los mamíferos tienen pulmones	$m \to p$
Por tanto, todos los murciélagos tienen pulmones	$u \to p$
2	
Todas las truchas son mamíferos	$t \to m$
Todos los mamíferos tienen alas	$m \to l$
Luego, todas las truchas tienen alas	$t \to l$

Tabla A.3: Argumentos válidos con premisas verdaderas y falsas, y sus conclusiones respectivas. Adaptado de [1].

A continuación, se menciona una manera de comprobar la validez de un argumento, pues es esta posibilidad de la Lógica, la de contar con un mecanismo de comprobación, la que nos interesa para construir agentes racionales: cuyas acciones sean consecuencia lógica de su ambiente y sus percepciones. Para ello es conveniente mencionar los conceptos que a continuación se anotan.

- Forma Argumental: arreglo de símbolos con variables sentenciales (letras que substituyen a un enunciado con estricta consistencia). Comúnmente se usan $p, q, r \ldots$, etc.
- Instancia de substitución: sustitución de enunciados por variables (les da un contenido).
- Forma específica: forma argumental de un argumento una vez que se ha instanciado con enunciados diferentes.
- Refutación por analogía lógica: si puede mostrarse que la forma específica de un argumento dado tiene una instancia de sustitución con premisas verdaderas y conclusión falsa, entonces, el argumento es inválido.

De esta manera:

> una forma argumental inválida tiene cuando menos una instancia de substitución con premisas verdaderas y conclusión falta. Toda forma argumental que no sea inválida, es válida, esto es, no tiene premisas verdaderas y conclusión falsa en ninguna instancia de substitución.

Como es evidente, para probar si una forma argumental es válida, habrá que verificar todas las instancias de substitución posibles, por ejemplo, mediante una tabla de verdad, en la que cada renglón representa una clase completa de instancias de substitución, donde los valores de verdad se representan por (F o V) o (1 o 0) y están en función de los enunciados que pueden sustituirse por las variables [1]

Esta revisión exhaustiva plantea el peligro de una explosión combinatoria, afortunadamente existen otras maneras de verificar la validez de un argumento, mismas que se tocarán en las siguientes partes de estos apuntes, por el momento, conviene revisar el método basado en tablas, que, se insiste, revisa exhaustivamente las posibilidades de las instancias de sustitución de un argumento dado.

En lo sucesivo muestra dos ejemplos de verificación de validez o invalidez de dos formas argumentales, a saber: si la forma específica de una argumento puede mostrarse que tiene una instancia de substitución con premisas verdaderas y conclusión falsa, el argumento es inválido. Véase esto en la tabla A.4.

Proposiciones	Forma argumental	Tablas de verdad
1		
Si soy presidente, entonces soy famoso	$p \to f$[2]	
Yo no soy presidente	$\neg p$	
Por lo tanto, yo no soy famoso	$\neg p$	

p	f	$\neg p \vee f$	$\neg p$	$\neg f$
1	1	*1*	*0*	*0*
1	0	*0*	*0*	*1*
0	1	**1**	**1**	**0**
0	0	*1*	*1*	*1*

Este argumento pareciera válido porque su conclusión es verdadera, sin embargo, obsérvese la tabla de verdad y compárese con el siguiente argumento con igual forma argumental, pero diferente contenido:

2		
Si Doroteo Arango es presidente, entonces es famoso	$p \to f$	
Doroteo Arango no es presidente	$\neg p$	
Luego, Doroteo Arango no es famoso	$\neg p$	

p	f	$\neg p \vee f$	$\neg p$	$\neg f$
1	1	*1*	*0*	*0*
1	0	*0*	*0*	*1*
0	1	**1**	**1**	**0**
0	0	*1*	*1*	*1*

Tabla A.4: Se muestra que el argumento es falso si se observa la tabla de verdad, en el tercer renglón se puede ver que las premisas son verdaderas y falsa su conclusión. Esto es suficiente para considerar el argumento como inválido. En el segundo caso es más evidente esta situación, pero se actúa de la misma manera, examinando la tabla de verdad.

Por otro lado, en la figura A.1 se muestran argumentos válidos: el silogismo disyuntivo y las reglas de inferencia *Modus Ponendo Ponens* y *Modus Tollendo Tollens* (Modo de afirmar afirmando y negar negando, respectivamente), mismas que se retomarán posteriormente, y que son relevantes por su aplicación en los métodos de razonamiento, por encadenamiento hacia adelante y hacia atrás, que se verán posteriormente en estos apuntes. En los renglones resaltados en color se muestra el caso donde las premisas son verdaderas y la conclusión también lo es, garantizando con ello su validez, pues no puede tener premisas verdaderas y conclusión falsa.

Finalmente, hay que tomar en cuenta que la Lógica tiene todo un sistema axiomático deductivo, para hacer la clasificación de argumentos válidos e inválidos. De tal suerte, cuenta con axiomas, leyes y teoremas que ayudan a obtener consecuencias a partir de ciertas premisas o a verificar si una consecuencia está

Forma argumental		Tabla de verdad

Modus Ponens

Si p, entonces q	p → q
Está dado p	p
Por lo tanto, q	q

p	q	¬p v q	p	q
1	1	**1**	**1**	**1**
1	0	**0**	**1**	**0**
0	1	**1**	**0**	**1**
0	0	**1**	**0**	**0**

Modus Tollens

Si p, entonces q	p → q
No está dado q	¬q
Luego, se niega p	¬p

p	q	¬p v q	¬q	¬p
1	1	**1**	**0**	**0**
1	0	**0**	**1**	**0**
0	1	**1**	**0**	**1**
0	0	**1**	**1**	**1**

Silogismo Disyuntivo

Si p o q	p v q
No está dado p	¬p
Luego, es cierto q	q

p	q	p v q	¬p	q
1	1	**1**	**0**	**1**
1	0	**1**	**0**	**0**
0	1	**1**	**1**	**1**
0	0	**0**	**1**	**0**

Figura A.1: Demostración de la validez del Silogismo disyuntivo, y las reglas de inferencia simple Modus Ponendo Ponens (Modus Ponens) y Modus Tollendo Tollens (Modus Tollens).

derivada lógicamente de premisas válidas. Se requiere de tres elementos para conformar un sistema formal, un sistema axiomático deductivo, los cuales se enuncian enseguida:

1. Un alfabeto de símbolos
2. Reglas de formación
3. Axiomas y reglas de inferencia

De esta manera, los dos primeros elementos nos dicen si una expresión pertenece al lenguaje, es decir, si es una **fórmula bien formada (fbf)**, en otras palabaras, nos indica cuándo una expresión está bien escrita o construida, el tercero es la base para hacer nuevas expresiones válidas y es el mecanismo por el cual se construyen razonamientos dentro del sistema. La lógica es un sistema formal, por ello, cuenta con estos elementos. A continuación, en la tabla A.5 se hace mención de ellos, tomando el caso de la **lógica de primer orden** o **de predicados**. Entre estas leyes se encuentran las que enseguida se mencionan [67, 66].

- Leyes o principios del pensamiento
- Leyes de Morgan
- Leyes o axiomas de determinación de tautologías

Es a partir de estas leyes y axiomas, y las que se enuncian en la tabla A.6, que la lógica confiere el poder expresivo para representar fácil y fiablemente el conocimiento y, se insiste, puede verificar si un razonamiento es válido o inválido.

La Lógica es de gran utilidad, empero, no es el único formalismo que se ha propuesto o usado dentro de la IA, de tal manera, en la siguiente sección se mencionan otros enfoques para hacer representación del conocimiento: estructuras de datos.

Elemento	Descripción
Alfabeto	Variables $(X, Y, Z \ldots)$, Constantes $(a, b, c \ldots)$, símbolos para denotar functores y predicados –o relaciones– con cierta aridad $(p(a), f(X) \ldots)$, conectivos –u operadores– lógicos ($\vee$ disyunción, $\wedge$ conjunción, $\neg$ negación, $\rightarrow$ implicación, $\Leftrightarrow$ doble implicación), símbolos de cuantificación ($\forall$ universal y $\exists$ existencial), otros símbolos, como paréntesis ()
Reglas de formación de términos	▪ Constantes y Variables son términos ▪ Cualquier expresión $f(t_1, \ldots, t_n)$ de $n \geq 1$ argumentos (donde cada argumento t_i es un término y f es un functor de aridad n) es un término. Sus variables libres son las variables libres de cualquiera de los términos t_i.
"Principios del pensamiento"	1. Identidad $(p \rightarrow p)$: si algún *es verdadero*, entonces *es verdadero*. 2. No contradicción ($p \wedge \neg p$ es falso): ningún enunciado puede ser *verdadero y falso*. 3. Tercero excluido ($p \vee \neg p$ es verdadero): todo enunciado es *verdadero o falso*.
Axiomas básicos	1. Doble negación $\neg\neg p \equiv p$: la negación de la negación de un enunciado es equivalente al enunciado no negado. 2. $p \rightarrow (q \rightarrow p)$: si p es verdadero, entonces se sigue de cualquier enunciado (en este caso q 3. $(p \rightarrow (q \rightarrow r)) \rightarrow ((p \rightarrow q) \rightarrow (p \rightarrow r))$: para un enunciado r que es implicado por otro q que a su vez es implicado por un primero p, entonces tanto q como r son implicados por p 4. $(\neg p \rightarrow \neg q) \rightarrow (q \rightarrow p)$: un enunciado falso implica a cualquier otro 5. $(p \vee q) \rightarrow (q \rightarrow p)$: para cualquier par de enunciados p y q, al menos uno de los dos implica al otro

Tabla A.5: Elementos de la lógica proposicional: alfabeto, reglas de formación, axiomas

Leyes o tautologías

- Conmutatividad: $p \vee q \Leftrightarrow q \vee p$ y $p \wedge q \Leftrightarrow q \wedge p$
- Asociatividad: $(p \vee q) \vee r \Leftrightarrow p \wedge (q \vee r)$ y $(p \wedge q) \wedge r \Leftrightarrow p \wedge (q \wedge r)$
- Distribuitividad: $(p \vee q) \wedge r \Leftrightarrow (p \wedge r) \vee V(q \wedge r)$ y $(p \wedge q) \vee r \Leftrightarrow (p \vee r) \wedge (q \vee Vr)$
- Idempotencia: $(p \vee p) \Leftrightarrow p$ y $(p \wedge p) \Leftrightarrow p$
- Identidad: $(p \vee F) \Leftrightarrow p, (p \vee V \Leftrightarrow V, (p \wedge F) \Leftrightarrow F$ y $(p \wedge V) \Leftrightarrow p$
- Complemento: $\neg V \Leftrightarrow F, \neg F \Leftrightarrow V, p \vee V \neg p \Leftrightarrow V$ y $p \wedge \neg p \Leftrightarrow F$
- Leyes de de Morgan: $\neg(p \vee q) \Leftrightarrow \neg p \wedge \neg q$ y $\neg(p \wedge q) \Leftrightarrow \neg p \vee \neg q$
- Absorción: $(p \vee q) \wedge q \Leftrightarrow q$ y $(p \wedge q) \vee q \Leftrightarrow q$
- Simetría de la equivalencia: $(p \Leftrightarrow q) \Leftrightarrow (q \Leftrightarrow p)$
- Caracterización de la implicación: $(p \rightarrow q) \Leftrightarrow \neg p \vee q$
- Caracterización de la equivalencia: $(p \Leftrightarrow q) \Leftrightarrow [(p \rightarrow q) \wedge (q \rightarrow p)]$
- Contrarrecíproco: $(p \rightarrow q) \Leftrightarrow (\neg q \rightarrow \neg p)$
- Transitividad: $[(p \rightarrow q \wedge (q \rightarrow r)] \rightarrow (p \rightarrow r)$ y $[(p \Leftrightarrow q) \wedge (q \Leftrightarrow r)] \rightarrow (p \Leftrightarrow r)$

Tabla A.6: Algunas de las leyes más conocidas de la lógica proposicional

2. Estructuras de Datos

A primera vista, un *Tipo de Dato* (TD), una *Estructura de Datos* (ED) y un *Tipo de Dato Abstracto* (TDA) son lo mismo, sin embargo son distintos como se verá a continuación.

Un TD es una forma de representar datos dentro de un lenguaje de programación, también se les conoce como *tipos primitivos* y pueden contener valores distintos:

- Booleano: falso o verdadero.
- Numérico (entero o de punto flotante): almacenan cifras hasta un determinado tamaño de memoria.
- Alfabéticos: almacenan representaciones de caracteres de los alfabetos y algunos otros (como el código ASCII). Pueden ser unitarios o conformar varios de ellos (cadenas o textos).

Por su parte, los Tipos de Datos abstractos (TDA) son una forma de organizar los datos (que han de convertirse, tras una serie de procesamientos, en

información) dentro de un programa de computadora. Los TDA son entonces estructuras y operaciones que auxilian en el almacenamiento y manipulación de datos en grandes cantidades.

Finalmente, las ED, son las representaciones de los TDA concretas en un lenguaje de programación y dependen de los datos primitivos y operadores que el lenguaje tenga disponibles.

Las ED son determinantes para que los programas puedan realizar operaciones con los datos de una forma eficiente, ordenada y eficaz.

De una manera más formal podría pensarse en las ED/TDA en la anterior definición de los elementos de la lógica proposicional de la sección anterior. De esta forma, las ED/TDA son un modelo matemático que se compone de una serie de operaciones definidas en el propio modelo [68].

En realidad, las ED /TDA son generalizaciones de los tipos de datos primitivos manejados por una computadora a través de los distintos lenguajes (enteros, flotantes, caracteres, etc.) [68]. De igual manera, sus operaciones generalizan las operaciones primitivas (adición, substracción, división, encapsulamiento y generalización, entre otras). Una ED/TDA se puede, una vez definida, utilizar como un tipo de datos primitivo, a través de las variables o conjuntos de ellas. Algunas ED pueden contener datos primitivos homogéneos o heterogéneos. En muchos lenguajes de programación de alto nivel, las ED utilizan los mecanismos de apuntadores o punteros para tener acceso a la memoria dispónible para el programa de forma directa.

Dentro de las ED más conocidas se encuentran las siguientes:

- Arreglos: secuencias contiguas de celdas de memoria denotadas por un mismo identificador. Pueden ser unidimensionales o conformar matrices de múltiples dimensiones.

- Listas enlazadas: suponen una mejora a los arreglos en el sentido de su flexibilidad (crecer o decrecer según se requiera), sus elementos son accesibles y se pueden agregar o suprimir de la lista a voluntad. En su versión básica se pueden recorrer sólo en un sentido de principio a fin. Pero puede haber distintas variantes, como las listas doblemente enlazadas y las listas de listas, además de las que se mencionan a continuación.

- Colas: es una lista o arreglo que define reglas especiales de operación. Mantiene un elemento frontal y un elemento final bien indicados y ante nuevos datos, su inserción o supresión depende de tales elementos: la política de inserción se define como *primeras entradas, primeras salidas*. Esto es, el elemento en llegar primero marca la precedencia para su uso, tal como en una cola de espera para un transporte o un servicio en ventanilla de atención.

- Pilas: análoga a las Colas, las Pilas definen un elemento tope (último en haber sido insertado) y un fondo (primer elemento insertado). De esta forma, su política de inserción o supresión se basa en *primeras entrads, últimas salidas* y Pilas

- Árboles: son estructuras que asocian múltiples elementos (nodos) entre sí de forma jerárquica. Al elemento inicial se le denomina raíz y dependiendo del tipo de política de asociación puede tener múltiples elementos que partan de él (denominados hojas). A su vez, cada hoja deviene en rama, es

decir, puede hacer las veces de elemento raíz (denominado nodo interno), y formar así una ramificación al asociar múltiples nodos dependientes (que a su vez son nodos internos u hojas).

- Grafos: son estructuras que asocian múltiples elementos también, pero sin la restricción jerárquica de los árboles. Son así una generalización de ellos, o si se prefiere, los árboles son una forma restringida por la jerarquía de los grafos.
- Conjuntos: son colecciones de elementos que pueden ser elementos primitivos u otras ED, incluyendo conjuntos. Todos sus miembros son distintos, se dice que son atómicos si son elementos primitivos. Tienen las típicas operaciones de unión, intersección, suprimir, insertar, membresía, entre otras. Pueden implementarse mediante arreglos o listas.
- Diccionarios: son formas más restringidas de conjuntos, que eliminan operaciones como la unión o la intersección.
- Tablas de dispersión: son formas de implementar diccionarios, que eficientan el acceso a los datos con el objetivo de que se mantenga el mismo tiempo para cada operación.

Apéndice B

Breve manual de NetLogo

Pasos clave para programar en NetLogo.

1. Tener el modelo diseñado.
2. Definir el ambiente (dar funcionalidad y propiedades a los `patches` o parcelas.
3. Definir los agentes que contendrá (`turtles`, `links`). Al inicio no hay agentes pero tanto el observador como las parcelas pueden crearlos.
4. Definir entradas, salidas, el inicio y fin de la simulación.

1. El ambiente de desarrollo NetLogo

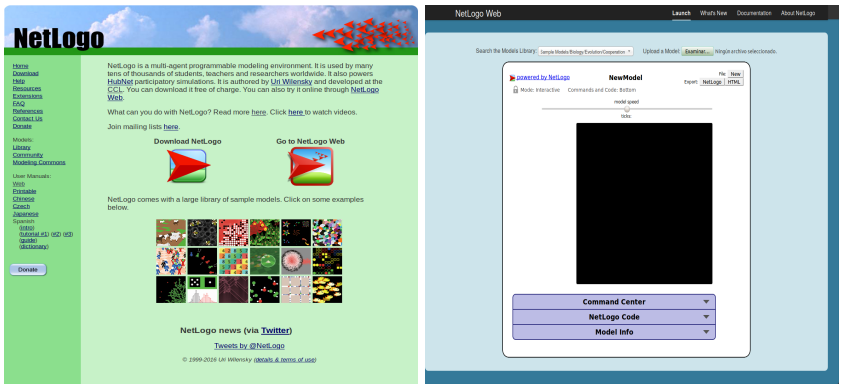
En esta sección se da una brevísima introducción a modo de pasos para introducirse en la programación en NetLogo. En primer lugar hay que entender su composición como entorno de desarrollo de SMA y, por ende de MBA.

De esta forma, tenemos los siguientes puntos a realizar:

1. Lo primero que hya que hacer es descargar NetLogo de su sitio de distribución oficial.
2. Interactuar con una simulación, conocer los elementos del ambiente, tales como:
   - Solapas *Ejecutar*, *Información* y *Código*
   - Entender los datos de salida
   - El espacio de comportamientos
3. Revisar la documentación clave, ayuda y recursos útiles:
   - Biblioteca de modelos
   - Guía de usuario
   - Ayuda en línea
   - Libros de texto y artículos

Para descargar NetLogo se debe acudir a los sitios electrónicos oficiales:

- Sitio web de descarga del Centro de Aprendizaje Conexo y Modelado Basado en Computadora, o CCL por sus siglas en inglés: *Center of Connected Learning and Computer-Based Modeling*)
  https://ccl.northwestern.edu/netlogo/download.shtml
- Ahí se puede descargar el entorno a través de paquetes interplataforma: MAC, Windows, GNU/Linux.
- O si se prefiere también se puede usar en línea: https:http://www.netlogoweb.org/launch
- Esto se puede ver en la figura B.1



a) b)

Figura B.1: a) Sitio oficial de descarga de NetLogo; b) su entorno en línea.

1.1. Elementos del entorno de NetLogo

Como NetLogo es una plataforma gráfica de programación, conviene conocer sus elementos:

- Una barra de tres solapas contiene las partes principales:
  1. **Ejecutar**: es la interfaz principal, contiene todos los elementos de control, visualización de la ejecución y la salidas solicitadas (gráficas, interruptores, botones, texto, etc.).
  2. **Información**: documenta el proyecto como respuestas a varias preguntas y secciones: ¿qué es esto?, ¿cómo usarlo?, autores, etc.
  3. **Código**: un editor donde se programa en lenguaje NetLogo el funcionamiento de los distintos componentes de la simulación.
- Además, una barra de menú con las opciones usuales y una barra de controles bajo las solapas.

Las solapas principales del entorno son:

- Comprobar: verifica errores de sintaxis.
- Buscar: busca cadenas en el código.
- Procedimientos: navega entre los archivos asociados al programa actual (modularidad).

En la figura B.2 se puede ver lo anterior.

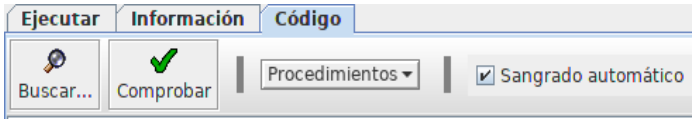


Figura B.2: Solapas del entorno de NetLogo.

Las simulaciones se visualizan en la sección de *Ejecutar* que al mismo tiempo define la interfaz:

- La interfaz principal, donde se ejecuta el modelo.
  1. Incluye todas las herramientas para construir el modelo e inspeccionarlo.
  2. Define el ambiente y sitúa a los agentes en él.
  3. Inicialmente vacía, se pueden editar (añadir, borrar, cambiar) los elementos según se requiera.
  4. Incluye una barra de comandos, para interactuar con los agentes.

La interfaz completa del ambiente se puede ver en la Figura B.3, donde se aprecia el entorno donde los agentes realizarán sus comportamientos, es decir, la rejilla 2D compuesta por los *parches* que la pueblan y que pueden a su vez tener distintos comportamientos, a excepción de movimiento.

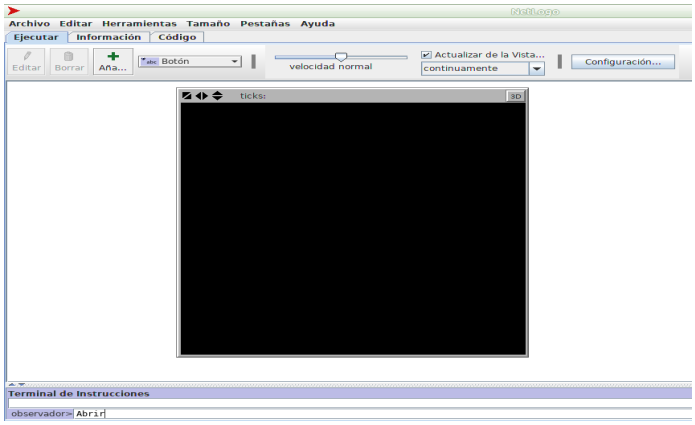


Figura B.3: Interfaz de NetLogo.

Los elementos del ambiente *Menú principal*, *Barra de control* y *Línea de comandos* se ilustran en la figura B.4. El *Menú principal* permite acceder a las

opciones de manejo de archivos, edición del archivo actual, y a herramientas de edición y configuración. En tanto que la *Barra de control* permite definir la velocidad de reporducción y el acceso a añadir elementos de intefaz así como a la configuración del entorno en general. Por otro lado, la línea de comandos permite al usuario insertar instrucciones en el lenguaje de NetLogo para que se ejecuten durante la simulación o previo a que ésta esté en funcionamiento.

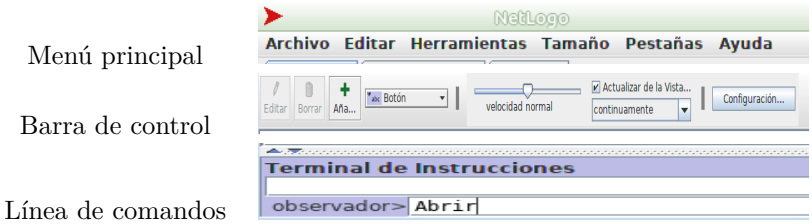


Figura B.4: Elementos de la interfaz de NetLogo.

El entorno se puede configurar de distintas formas. Por ejemplo, para configurar el ambiente se utiliza el Botón *Configuración*, que define el aspecto del ambiente, la ubicación del origen, el máximo de elementos (recordar que es una rejilla) en el eje X y en el Y, si el mundo tendrá un límitne horizontal y/o vertical, el tamaño de la parcela (la dimensión de cada elemento de la rejilla) la tasa de despliegue de la simulación, y el contador temporal, tal como se muestra en la figura B.5.

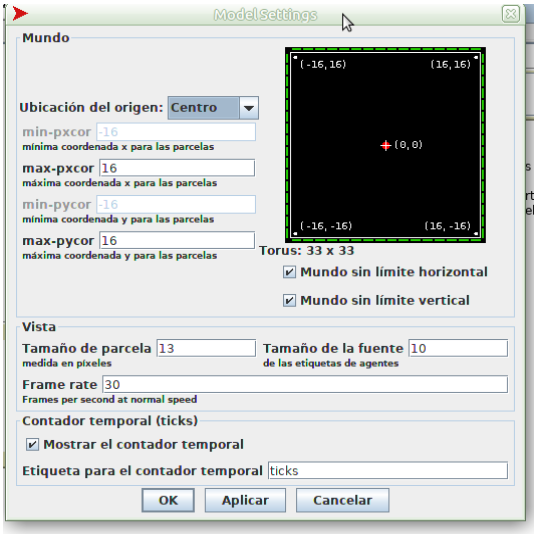


Figura B.5: Botón de configuración.

1.2. Controles de entrada

Botones

Los bontones permiten detonar procedimientos y funciones, por ejemplo en la figura B.6 se muestra cómo crear un botón, junto con la caja de control que asocia su accionar con un conjunto de instrucciones a ser ejecutadas.

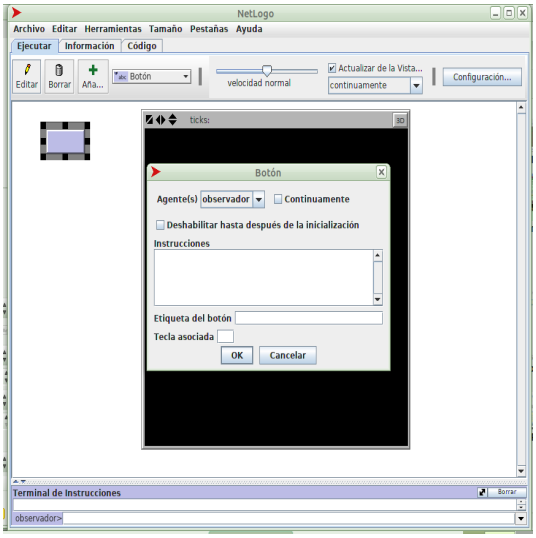


Figura B.6: Creación de botones para ejecutar tareas.

Una vez creado el botón, se debe asignar una funcion o un procedimiento (representado en el código del programa), como se ve en la figura B.7, donde se le está asignando la detonación del procecimiento *setup*, el cual debe estar implementardo en el código.

También se puede, desde luego, cambiar el nombre a los botones que se han definido, esto se ve en la figura B.8.

Interruptores, deslizadores y diales

El entorno permite además añadir interruptores y diales a la interfaz. Estos elementos permiten añadir entradas del usuario que modifican directamente los valores de ciertas variables globales de los agentes y otros elementos de la simulación:

- Por posición (interruptor: activado/desactivado), ver Figura B.9.
- Por graduación (dial o deslizador), ver Figura B.10,donde se añade dicho elemento a la interfaz, la figura B.11 donde se muestra cómo configurarlos, y la figura B.12 que muestra cuando ya se ha aceptado la configuración del deslizador.

Captura de datos

También se puede configurar la interfaz para aceptar datos proporcionados por el usuario, como se ve en la figura B.13

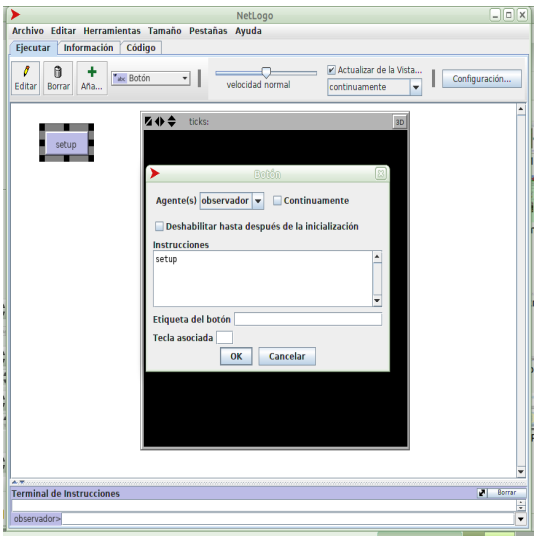


Figura B.7: Asignarle una función/procedimiento: `setup`.

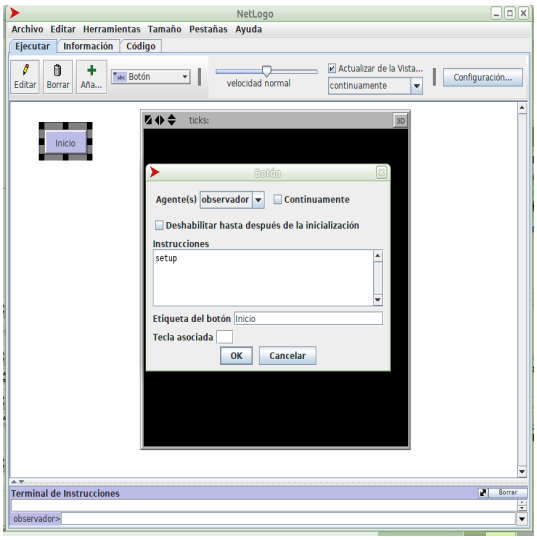


Figura B.8: Cambiar el nombre del botón a `Inicio`.

En la figura B.14 se muestra cómo añador entradas textuales.

1.3. Controles de salida

El ambiente de NetLogo también proporciona elementos para controlar la salida con el objetivo de proveer de información al usuario sobre el estado de la simulación. Esto se aprecia en la figura B.15.

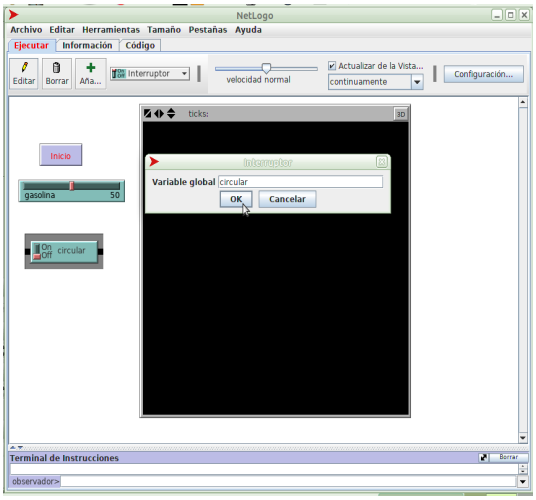


Figura B.9: Interruptor de entrada.

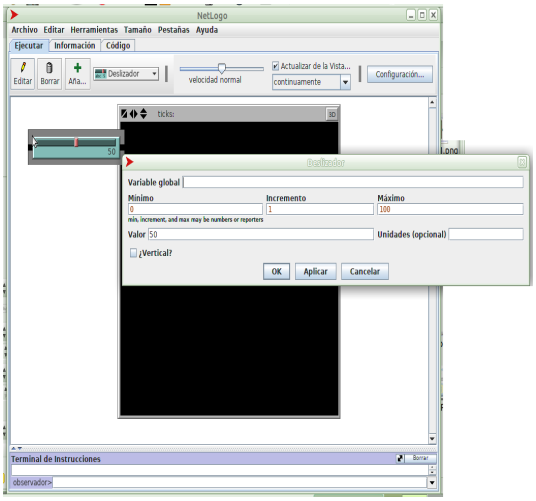


Figura B.10: Añadir deslizadores.

Trazos de variables

También se puede añadir graficadores en tiempo real para ver el funcionamiento de variables o expresiones dentro de la operación de la simulación (ver Figura B.16), como la caja de salida que se ve en la figura B.17.

Etiquetas

Y se puede además, añadir notas explicativas para proporcionar mayor información sobre la simulación, sus parámetros y funcionamiento.

Esto se puede realizar como se ve en la figura B.18.

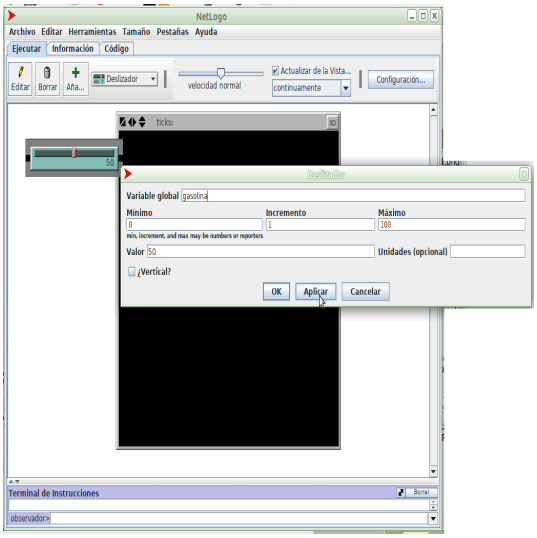


Figura B.11: Configurar deslizadores.

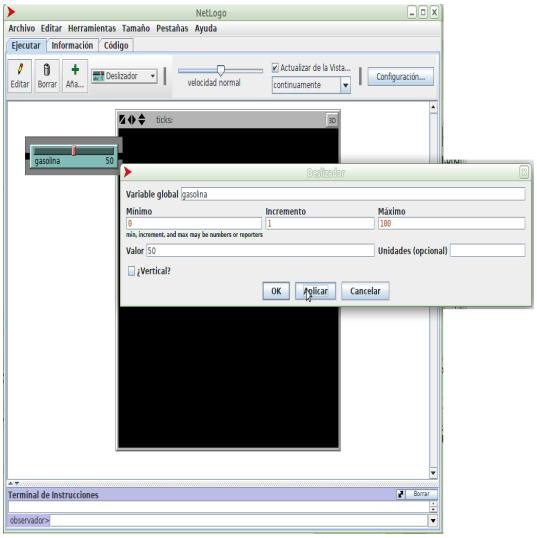


Figura B.12: Aceptar configuración del deslizador.

Componentes de entrada/salida.

Finalmente, en la figura B.19, se ve cómo luce la entrada y salida en el ambiente de NetLogo.

1.4. El lenguaje de programación

Lo primero que hay que entender es que tiene dos componentes básicos de comportamiento:

- Tortugas (del inglés *Turtles*): son *agentes reactivos* que simulan personas,

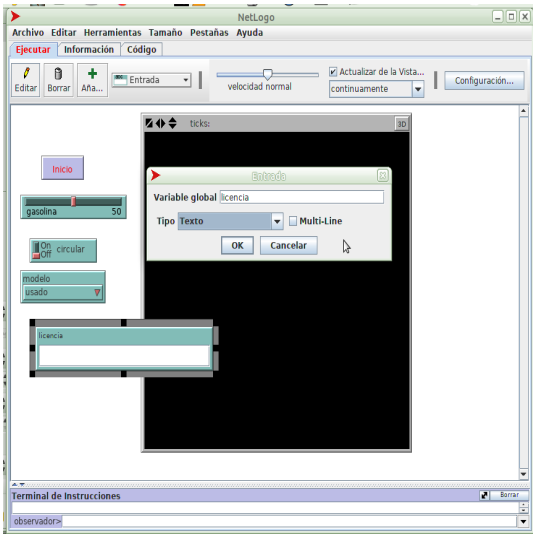


Figura B.13: Configurar entrada [número, texto, color, funciones].

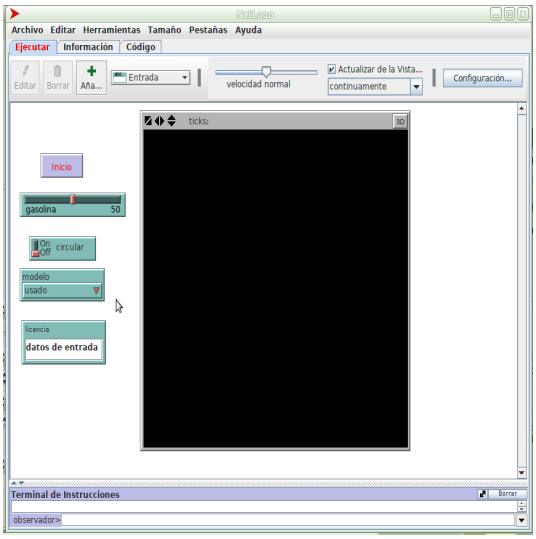


Figura B.14: Añadir entrada de texto.

grupos, instituciones, etc. Son los agentes "normales" de NetLogo

- Tiene propiedades y variables asociadas a ellos.
- Pueden dividirse en razas ("breeds"):
  `breed [sapos sapo]`: define agentes tipo "sapo".
  `sapos-own [color veneno]`: define dos variables (`color` y `veneno`) sólo para la raza `sapos`.

- *Parches* (del inglés *Patches*): representan unidades de territorio o del ambiente, también poseen propiedades modificables como los agentes.

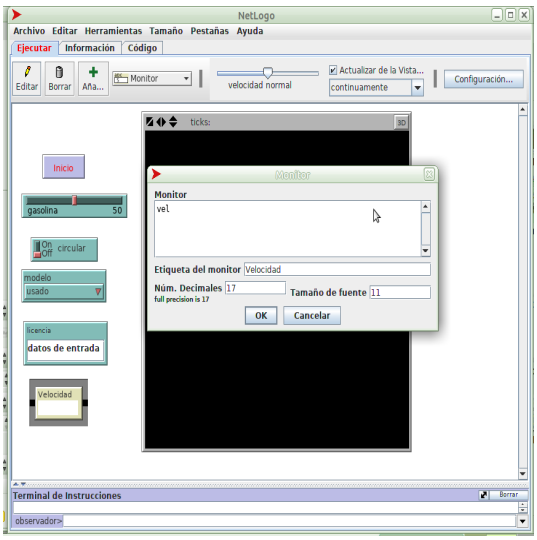


Figura B.15: Añadir monitor

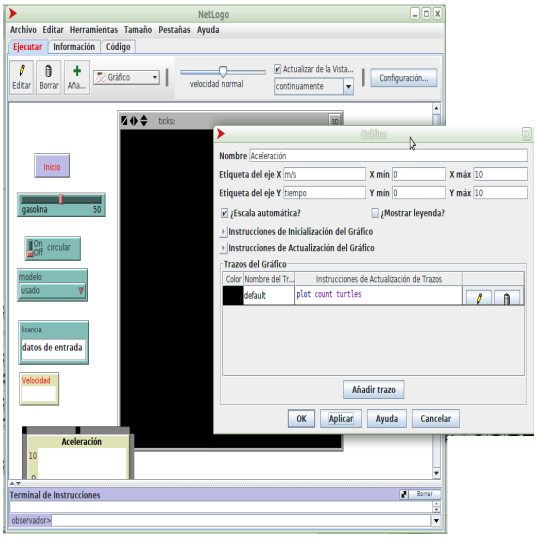


Figura B.16: Añadir graficador (tiempo real) de variables o expresiones.

- *Ligas o enlaces*: representan el vínculo entre dos agentes.

Características del lenguaje de programación NetLogo:

- Su lenguaje de programación es un dialecto del Logo:
  `ask turtles[fd 1]`: solicita a los agentes avanzar un paso.
- Es un lenguaje semifuncional basado en el procesamiento de listas y aplicaciones de funciones a éstas.
  `[elem1 elem2]`.
- Realiza asignación imperativa y soporta modularidad al incluir primitivas

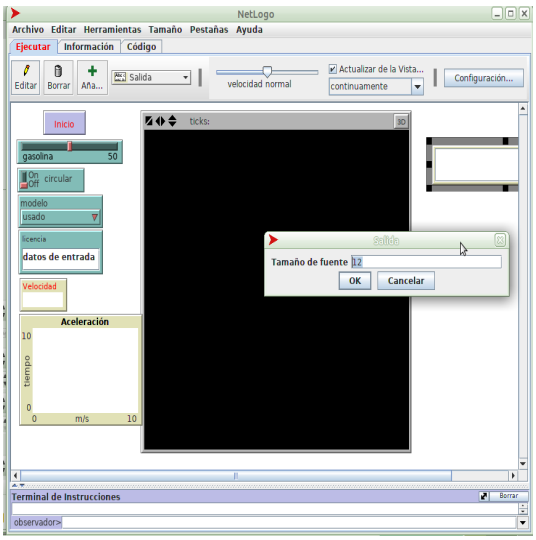


Figura B.17: Añadir graficador (tiempo real) de variables o expresiones.

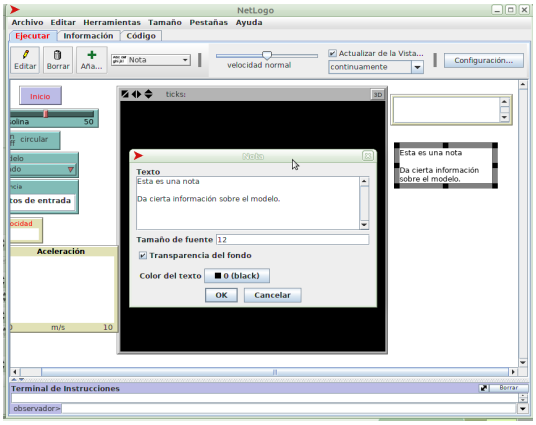


Figura B.18: Añadir nota explicativa.

nativas y definición de procedimientos y funciones, además de la inclusión de otros archivos de código: `_includes [ nombre-de-archivo ... ]`.

Funciones y procedimientos nativos (primitivas):

- Amplia gama primitivas:
  `show mean [edad] of turtles`: calcula el promedio de la variable `edad` de todos los agentes.
  `show max [ingreso] of desarrolladores`: calcula el máximo de la variable `precio` de todos los agentes de tipo `desarrolladores`.

- Procedimientos (solicita a los agentes una tarea):

```
to corre
ask turtles [fd 1]
end
```

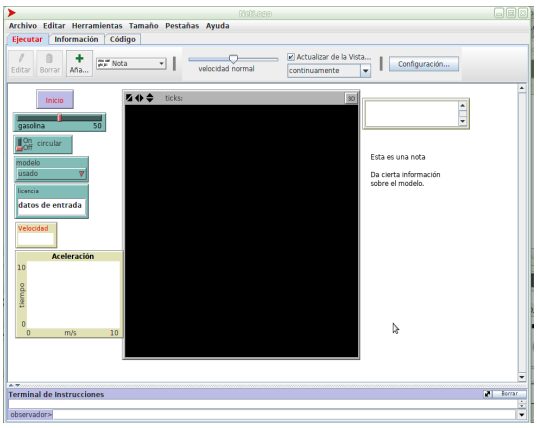


Figura B.19: Entrada y Salida en el entorno de NetLogo.

- También pueden tener parámetros: `to corre [a b] ...` , invocación: `corre a b`
- Reporteadores o funciones (devuelven un valor de variable o cálculo).

```
to-report preg?
return var
end
```

- También con parámetros: `to-report preg? [a b] ...` , Invocación: `preg? a b`

2. Código del modelo Influencia Social

Aquí se muestra el correspondiente código de la funcionalidad básica del modelo Influencia Social, no se muestra el código correspondiente a la interfaz, pero éste se puede obtener en la dirección siguiente: http://lab-ltsi.cua.uam.mx/Docencia/LibroIntroSMA-1/Codigos/.

```
 ;;
;; Wulfrano Arturo Luna Ramirez
;;
;; Influencia social
;;

;; Variables globales (comunes a todos los agentes)
;; forma se define el el controlador de tipo seleccionador "forma"
;; poblacion se define el el controlador de tipo deslizador "
    poblacion"
globals [verdes rojos]
turtles-own[cambios]

;; Procedimiento de inicio
to inicio
  clear-all
  reset-ticks
  set verdes poblacion / 2
  set rojos  poblacion / 2
  crear-agentes verdes rojos
  contar-vr
end

to avanzar
  dejar-huella
  correr
  contar-vr
  tick
end

;;Procedimiento crear-agentes
to crear-agentes [num-v num-r]
  creaAgente num-v green
  creaAgente num-r red
end

;;Procedimiento creaAgente
to creaAgente [num col]
  ;; crea num agentes con la forma man y el color indicado por col
  create-turtles num
  [set color col
   set shape forma
   fd 10 ]

end

;; Procedimiento correr
to correr
  ask turtles[
    right random 360
    forward 1
  ]
   cambiar
end
```

```
54
55 ;; Procedimiento cambiar
56 to cambiar
57   let r 0
58   let v 0
59 ask turtles[
60   ;; cuenta los agentes rojos alrededor
61   set r verdes-cerca
62   ;; cuenta los agentes verdes alrededor
63   set v rojos-cerca
64   ;; adopta el color de la mayoria
65   ifelse r > v [set color red] [set color green]
66   ]
67 end
68
69 ;; Procedimiento contar agentes verdes/rojos
70 to contar-vr
71   set rojos  count turtles with [color = red]
72   set verdes count turtles with [color = green]
73 end
74
75 to dejar-huella
76   if huella = true
77   [ask turtles [pen-down]]
78   if huella = false
79   [ask turtles [pen-up]]
80 end
81 ;;
82 ;;Funciones o reportadores
83 ;;
84 ;; reporta cuantos verdes hay cerca
85 to-report verdes-cerca
86   let g 0
87 ask turtles-on neighbors [set g count turtles with [color = green]]
88  report  g
89 end
90
91 ;; reporta cuantos rojos hay cerca
92 to-report rojos-cerca
93   let r 0
94  ask turtles-on neighbors [set r count turtles with [color = red]]
95  report  r
96 end
```

Código B.1: Código del modelo de Influencia Social `influenciaSocial.nlogo`

Apéndice C

Un esquema de trabajo con IAG

El auge de la Inteligencia Artificial Generativa (IAG) requiere mención especial, particularmente la IAG referida a los sistemas dialogantes o *chatbots*, de la mano de los generadores de imágenes. Tales herramientas tienen implicaciones para los procesos de enseñanza aprendizaje. Como herramientas disruptivas tanto pos sus característics técnicas como por su amplia disponibilidad y publicidad configuran tanto un reto como una oportunidad para los centros educativos, específicamente para las universidades.

1. ¿Usar la IAG?

A continuación se propone un marco de trabajo en el contexto educativo cuyo propósito es orientar sobre el uso que resulte en beneficio de los estudiantes y los aleje de falsas promesas y pretenciones derivadas de la publicidad y desinformación alrededor de las herramientas de la IAG.

Esta propuesta se yergue sobre el trabajo principalmente expuesto en [69, 70, 71, 72, 73], donde se establece la centralidad de ejercer metódicamente el sentido crítico tanto de docentes como de estudiantes al interactuar con herramientas de IAG y valorar sus posibilidades. Para ello, deben identificarse las fortalezas y debilidades de unos y otros. Por un lado, el docente debe mantener el control del proceso de enseñanza-aprendizaje y saber utilizar las herramientas como inspiración o apoyo para la generación de materiales didácticos. De esta forma se fomentará el análisis crítico y la resolución de problemas. Por otro lado, los estudiantes debe concientizarse de no tomar los resultados obtenidos a través de las herramientas de IAG como un producto final, sino como el punto de partida para posteriormente conseguir algo más relevante, veraz y certero. De esta forma, se trata de desarrollar dialécticamente el pensamiento, es decir:

1. los productos de la IAG son la *tesis*,
2. a ella de debe contrastar una *antítesis*,
3. para finalmente arribar a un *síntesis*.

Tras esto, se puede aspirar a obtener un producto encaminado a generar conocimiento (tal como el proceso dialéctico en sí mismo).

1.1. La IAG, ¿para qué?

Se pueden considerar algunos de los siguientes productos para emplear herramientas de IAG:

- Generación de contenido (ya sea texto o multimedia).
- Traducción de idiomas/lenguajes (programación de computadoras).
- Corrección o detección de errores.
- Además, debido a que con la IAG se pueden generar, en pocos segundos, materiales editables, los docentes podrían darle un uso a la IAG en torno a:
  - Generar lluvias de ideas o material de discusión
  - Automatizar la obtención de calificaciones y comentarios.
- Generar materiales iniciales como cuestionarios y explicaciones.
- Plantear problemas simples como base para su posterior desarrollo.

2. ¿Cómo usar la IAG?

El uso de la IAG en la docencia requiere que se asignen responsabilidades a docentes y alumnos. Se debe aplicar un método de razonamiento iterativo y crítico para valorar los productos obtenidos.

En la figura C.1 se muestra un esquema a forma de guía de uso de la IAG en la docencia con base en las tres fases de la dialéctica: tesis, antítesis y síntesis.

De este modo, una vez que se tiene un producto obtenido mediante IAG, se presentan las etapas a seguir [70]:

- Análisis: determinar qué es lo que se está obteniendo, identificar los rasgos relevantes para la tarea en cuestión del resultado.
- Verificación: identificar si lo que se obtuvo se sostiene, argumental, factual y referencialmente conforme el conocimiento académico y científico.
- Contrastación: buscar elementos probatorios o que tengan puntos de vista distintos, argumentos opuestos o experimentos de validación.
- Reescalamiento: pensar en una aplicación macro o micro del tema en cuestión.
- Profundización: ir a más detalles, más extenso o específico en el tema del que se trate.
- Reelaboración: con todo lo anterior, rehacer o replantear el resultado obtenido, enriquecido con la nueva información y sus contrastaciones, validaciones y modificaciones.

Así, se podrá utilizar una herramienta de IAG de una manera que beneficie el proceso de enseñanza-aprendizaje.

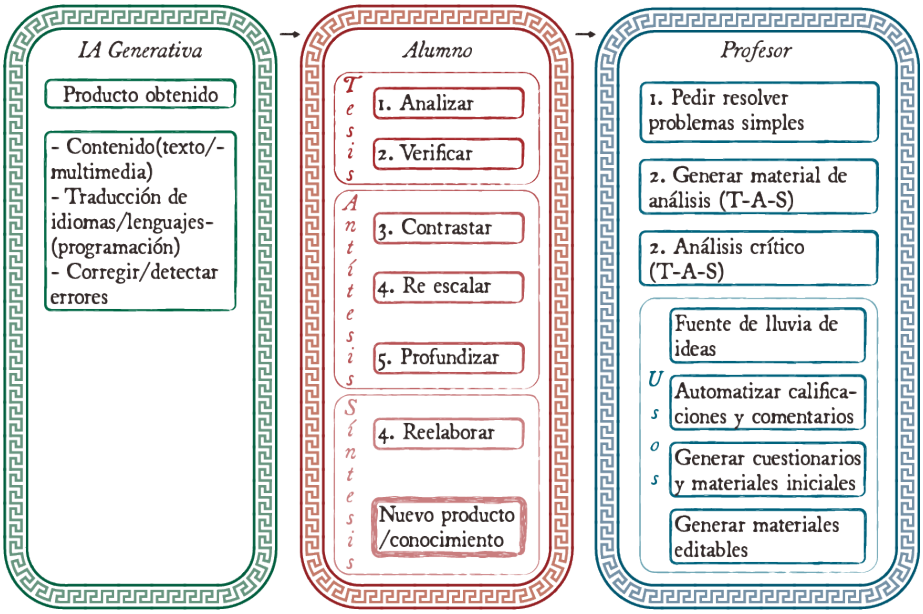


Figura C.1: Esquema de trabajo basado en la dialéctica para profesores y alumnos.

2.1. ¿Cuándo usar la IAG?

Por otro lado, los profesores podrían generar materiales y cuestionarios[1], ayudarse en la calificación, y proponer problemas orientados a que los alumnos transiten por las fases dialécticas y sus tareas antes mencionadas.

Finalmente, se presenta un árbol de decisión como guía para usar alguna herramienta de IAG, elaborado sobre la propuesta de Aleksander Tiulkanov [73]. Esto se muestra en la figura C.2. Para determinar el uso seguro de herramientas de IAG se debe incluir tanto el análisis de la veracidad buscada en el producto requerido, como una valoración de sus responsabilidades éticas y legales.

3. Conclusión

Conforme la guía antes mencionada, la IAG se podría utilizar para la creación de contenido textual (generar traducción de idiomas, resúmenes y ensayos breves) o de imágenes (composiciones o ilustraciones alusivas a un tema, o videos a partir de una descripción).

Los estudiantes deben estar al tanto de que luego habrán de pasar las etapas de análisis, verificación, contratación, reescalamiento, profundización y reelaboración del producto obtenido.

Por otro lado, los profesores podrían generar materiales y cuestionarios, ayudarse en la calificación, y proponer problemas orientados a que los alumnos transiten por las fases dialécticas y sus tareas antes mencionadas.

[1]Los cuestionarios de este libro fueron realizados parcialmente con una IA Generativa: Claude, conforme el esquema de trabajo propuesto en este apéndice.

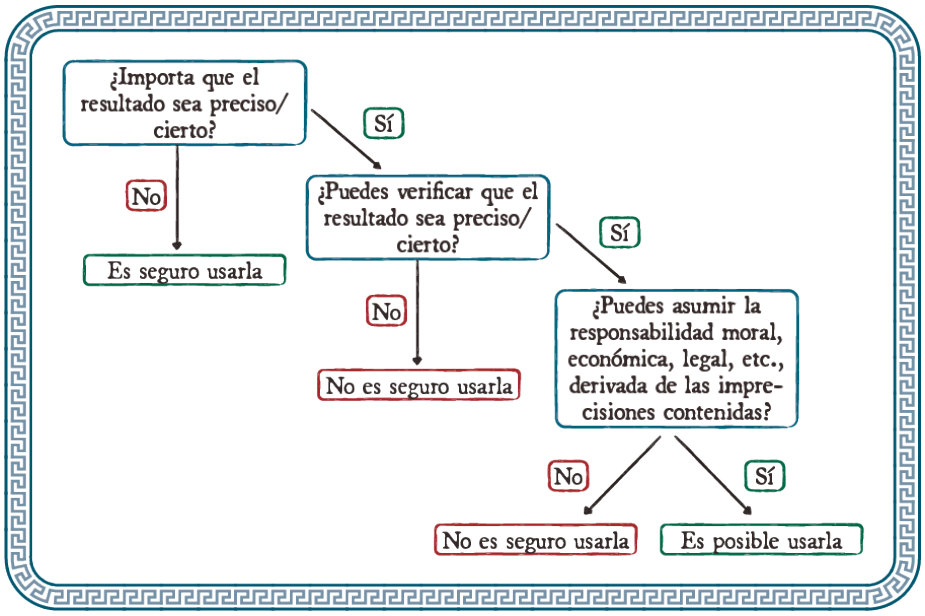


Figura C.2: Árbol de decisión para utilizar una herramienta de IAG.

Glosario

Ambiente determinista o estocástico: es determinista si los estados del medio están determinados por estado anteriores y la acción ejecutada por los agentes, en otro caso, es estocástico.

Ambiente discreto o continuo: depende del estado del medio, del manejo del tiempo, las percepciones y las acciones del agente

Ambiente episódico o secuencial: en el primer caso, la experiencia del agente ocurre en episodios atómicos (percepción + ejecución de una única acción posterior), y los episodios subsecuentes no dependen de las acciones de episodios pasados. Es secuencial cuando las decisiones actuales afectan las futuras.

Ambiente estático o dinámico: su dinamicidad depende de si cambia mientras el agente delibera.

Ambiente estratégico: es el medio determinista excepto para las acciones de otros agentes.

Ambiente monoagente o multiagente: depende de la evaluación de la medida de rendimiento, si para ello influyen las acciones de un solo agente o hay ingerencia de otros.

Ambiente total o parcialmente observable: depende de la capacidad de los sensores para tener acceso completo o parcial al estado del medio, esto es, a los detalles relevantes para la toma de decisiones

Ambiente semidinámico: si el ambiente no cambia, pero sí lo hace el rendimiento del agente.

Agente: define a un sistema computacional, tanto hardware como software situado en un ambiente, en el que se desenvuelve con el fin de conseguir sus objetivos de diseño.

Agente basado en modelo: almacena información de las partes del mundo que no son visibles en el momento actual, mantienen un estado interno para tener en cuenta la secuencia perceptual lo más completa posible. Necesita codificar la información sobre a) la evolución del mundo sin considerarse el mismo (modelo del mundo); y b) la manera en que afectan al mundo las acciones del agente.

Agente basado en objetivos: además del estado actual, guarda información sobre la meta que le ayude a discernir las situaciones deseables. Puede implicar mecanismo de búsqueda y planificación para determinar la selección de acciones con el fin de lograr sus metas. Es flexible debido a que su decisión se soporta con conocimiento representado explícitamente y es modificable.

Agente basado en una tabla: primera aproximacion de los agentes reactivos simples la cual tiene un mapeo directo entre percepciones y acciones. Este primer intento adolece rigidez (contraria a la autonomía buscada en los agentes) y una tabla exi giría muchos recursos en términos de diseño y búsqueda de

la solución, por lo cual es un intento condenado al fracaso como solución a la mayoría de los problemas.

Agente basado en reglas: otro intento son los agentes que emplean reglas de condición acción, del tipo: *SI < condición > ENTONCES < acción >*. Su gran problema es que sólo toman la decisión correcta al analizar la percepción actual, lo cual es aplicable sólo en entornos totalmente observables.

Agente basado en utilidad: incorpora una función de utilidad que proyecta una secuencia de estados (>= 1). Esta función permite decidir a) cuando hay objetivos conflictivos y sólo puede alcanzarse uno de ellos; y b) cuando hay varios objetivos que guíen al agente y no hay certeza de alcanzar ninguno, pondera la probabilidad de éxito en función de su importancia. Una función de utilidad explícita puede hacer que el agente tome decisiones racionales.

Agentes reactivos: agentes implementados como conjuntos de reglas si-entonces han sido utilizados extensamente para el desarrollo de Sistemas Multi-Agente en distintos dominios incluyendo simulaciones de desastres y como apoyo a tareas de salvamento y recuperación postevento.

Agente reactivo simple (agente reflejo): estos agentes seleccionan las acciones sobre la base de las percepciones actuales, sin tomar en cuenta la secuencia perceptual histórica.

Arquitectura de agente: se refiere al sustrato donde se ejecutará el programa de agente, que es la función que implanta el mapeo del agente.

Capacidad social: intercomunicación entre agentes y con otros sistemas.

Ciclo de razonamiento: implementa el proceso de toma de decisiones del agente. Se configura como un ciclo *sensado-deliberación-acción* o *percibir-decidir-incidir*, donde la toma de decisiones está implementada en la parte de selección de acción del agente.

Conocimiento: datos e información sobre el medio ambiente, sobre sí mismo y otras entidades.

Cuarto chino: objeción a la prueba de Turing propuestas por John Searle. La idea de la prueba es poner de manifiesto los aspectos cognitivos de la cuestión (estructura y operaciones internas que permiten que un sistema diga lo correcto –en tiempo y forma): una persona encerrada en un cuarto, sin saber nada de la lengua china, podría responder si tuviera a la mano un manual que relacionara frases eficazmente, pero no entendería nada.

Deliberación: razonar sobre sus percepciones y estímulos.

Dictum de Ada Lovelace: Una computadora sólo puede hacer lo que el programador le ha indicado, nada más.

Evento: n suceso significativo al que el agente debería responder de alguna manera se conoce como evento. Un evento puede ser extraído de las percepciones o generarse por la operación interna del ciclo de procesamiento del agente.

Estructura de agente: se dice que comprende Arquitectura + Programa de agente

Función de agente: abstracción determina el comportamiento de un agente (salida) a partir de su secuencia perceptual (entrada), es la parte relacionada con su racionalidad o deliberación.

Habilidad: acciones que el agente puede efectuar diestramente.

Inteligencia: es la capacidad de resolver problemas de orden intelectual, moral o vital en un tiempo determinado.

Inteligencia Artificial: es el estudio de la inteligencia natural por medios artificiales.

IA Clásica: postula que computar es pensar.

Nueva IA o IA Situada: busca que el mundo real sea el molde de la inteligencia de las entidades artificiales.

IA Hegemónica: sustentada en macrodatos, telemática y modelos estadísticos.

Juego de la imitación: plantea que dos personas sostengan una conversación por teletipo y una de ellas sea substituida por una computadora, –con las condiciones materiales para que la otra no se dé cuenta de la sustitución.

Meta: cuando un agente está trabajando en pro de algo, a ese algo se le denomina meta. Dependiendo de la arquitectura del agente, también puede denominarse tarea, objetivo o propósito o deseo.

Medioambiente: se entiende el entorno de trabajo donde el agente está situado, esto es, aquellos problemas para los cuales los agentes son una solución.

Medida de desempeño: define qué tan bueno es el agente, un estándar de logro de metas.

Neutralidad valorativa de la ciencia: que consiste en asumir como posible la supuesta limpieza ideológica de la ciencia, y pretender que ésta no tiene influencia en pro de los intereses de grupos empresariales, políticos o económicos.

Omnisciencia: un agente que tuviera esta facultad, podría estar siempre informado de todo su entorno y sabría qué resul tados tendrían sus acciones (todas y cada una), por ende actuaría escogiendo las que más beneficio le reportara. Esta característica es prácticamente imposible.

Percepción: es una sensación ya procesada, es decir, es la interpretación o clasificación de una sensación o grupo de ellas dentro de un determinado dominio. Así, cada sensación o grupo de ellas se asigna un significado relevante. En suma, es el resultado del procesamiento (interpretación) de una sensación.

Presciencia: es el caso en donde una gente conoce todo acerca del mundo y sus acciones presentes y futuras.

Problema: es toda aquella información que el agente necesita para actuar, pude considerarse como un conjunto formado por estados y acciones: $Problema = E, A$

Proactividad: ser persistente en el logro de sus objetivos.

Protocolo: designa a la definición de patrones de interacción válidos entre agentes (como la secuencia de mensajes en una comunicación).

Prueba de Turing: se basa en el juego de la imitación. Es una prueba para definir operativamente la inteligencia: una conducta es inteligente si logra una eficiencia asimilable a la del ser humano (una persona no la podría distinguir).

Racionalidad tecnológica (teórica): propia de la cuestión meramente operativa o instrumental, de funcionamiento técnico, ese sí fuera del alcance de las ideologías.

Racionalidad de los fines (práctica): referida a la cuestión de para qué se desarrolla algo, influíble totalmente por los fines y objetivos que persiguen los actores implicados.

Reactividad: capacidad de percibir cambios en su ambiente y poder responder a ellos.

Secuencia perceptual: todo lo que es percibido por el agente durante un lapso determinado.

Sensación: conjunto de señales y lecturas que un agente es capaz de recibir como entrada a través de sus dispositivos de entrada: cámaras, micrófonos,

dispositivos para detectar luces, vibraciones, etc.; es decir, el flujo de datos proveniente de sus sensores. Es lo que el agente ha recibido como entrada vía sus perceptores.

Sistemas Multi-Agente: la capacidad social de los agentes permite crear ensambles de ellos, los Sistemas Multi-Agente (SMA), que comparten su ambiente e interactúan para conseguir objetivos comunes Una característica importante de los SMA es su tolerancia a fallos y su control descentralizado, lo que los hace ideales para enfrentar distintas tareas con alto grado de robustez. Los SMA han sido aplicados a distintos dominios, desde el comercio electrónico y aplicaciones industriales, hasta simulaciones de fenómenos sociales y naturales. A su vez, un SMA puede estar compuesto por agentes homogéneos (similares en capacidades y diseño) o heterogéneos, que además de cumplir distintas tareas y poseer diferentes capacidades, son diseñados bajo los principios de distintas arquitecturas.

Simulación o Modelación Basada en Agentes (MBA): es una técnica computacional orientada a la representación y explicación generativa de sistemas complejos. Está fuertemente ligada a la idea del surgimiento emergente de comportamientos complejos durante el curso de la simulación, mediante esta se pueden observar comportamientos globales a partir de acciones simples de sus componentes individuales.

Sociabilididad: capacidad de interacción y comunicación con sus semejantes, otras entidades y el ambiente.

Bibliografía

[1] X. Ramón, *Introducción a la Historia de la Filosofía*. UNAM, 2000.

[2] S. Russell and P. Norvig, *Inteligencia Artificial, un enfoque moderno*. Pearson, 2004.

[3] M. Wooldridge and N. R. Jennings, "Intelligent agents: theory and practice," *The Knowledge Engineering Review*, vol. 10, pp. 115–152, 6 1995.

[4] N. M. José, "El Popol Vuh, ¿un experimento Gendanke?," in *De la Filosofía a la Inteligencia Artificial*, Megabyte, pp. 221–226, Noriega, 1992.

[5] M. Martínez-Morales, "La ciencia desde el Macuiltépetl: Del Popol Vuh a las máquinas inteligentes - Alef — alef.mx." http://alef.mx/la-ciencia-desde-el-macuiltepetl-del-popol-vuh-a-las-maquinas-inteligentes/, 2017. [Última visita 15 de noviembre de 2024].

[6] "El libro del consejo (Popol Vuh)," in *UNAM, Coordinación de Humanidades)* (P. D. F. T. y. n. d. G. R. González De Mendoza Y Miguel Ángel Asturias, J M, ed.), 1993.

[7] J. Haugeland, *La inteligencia artificial*. Siglo XXI, 2007.

[8] L. Álvarez Munárriz, *Fundamentos de inteligencia artificial*. Secretariado de Publicaciones, Universidad de Murcia, 1994.

[9] N. J. y. G. Edgar, *De la Filosofía a la Inteligencia Artificial*. Megabyte, 1992.

[10] B. D. Randall, *Intelligence as Adaptive Behavior. An Experiment in Computational Neuroethology*. 1990.

[11] R. A. Brooks, "Intelligence without representation," *Artificial Intelligence*, vol. 47, no. 1, pp. 139–159, 1991.

[12] T. Benn, "Weizenbaum's nightmares: how the inventor of the first chatbot turned against ai." https://www.theguardian.com/technology/2023/jul/25/joseph-weizenbaum-inventor-eliza-chatbot-turned-against-artificial-intelligence-ai, 2023. Accessed 10-05-2024.

[13] . G. A. Franklin, S., "Is it an agent, or just a program?," 1997.

[14] J. O. y Gasset, *Meditaciones del Quijote*. Residencia de estudiantes, 1914.

[15] J. M. Bradshaw, "An introduction to software agents," in *Software Agents* (J. M. Bradshaw, ed.), pp. 3–46, Cambridge, MA: AAAI Press/The MIT Press, 1997.

[16] H. S. Nwana and D. T. Ndumu, "An introduction to agent technology," in *Software Agents and Soft Computing: Towards Enhancing Machine Intelligence, Concepts and Applications*, (Berlin, Heidelberg), p. 3–26, Springer-Verlag, 1997.

[17] C. Hewitt, "Actor model of computation: Scalable robust information systems," 2015.

[18] H. S. Nwana, "Software agents: an overview," *The Knowledge Engineering Review*, vol. 11, no. 3, p. 205–244, 1996.

[19] M. R. Genesereth and M. L. Ginsberg, "Logic programming," *Commun. ACM*, vol. 28, p. 933–941, sep 1985.

[20] J. S. Rosenschein and M. R. Genesereth, "Deals among rational agents," in *Proceedings of the 9th International Joint Conference on Artificial Intelligence - Volume 1*, IJCAI'85, (San Francisco, CA, USA), p. 91–99, Morgan Kaufmann Publishers Inc., 1985.

[21] J. S. Rosenschein, "Multiagent systems, and the search for appropriate foundations," in *Proceedings of the 2013 International Conference on Autonomous Agents and Multi-Agent Systems*, AAMAS '13, (Richland, SC), p. 5–6, International Foundation for Autonomous Agents and Multiagent Systems, 2013.

[22] N. J. Nilsson, *Artificial Intelligence: A New Synthesis*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1998.

[23] M. Wooldridge, *An introduction to multiagent systems*. John Wiley & Sons Ltd, 2009.

[24] A. S. Rao and M. P. Georgeff, "Bdi agents: From theory to practice," in *In Proceedings of The First International Conference on Multi-Agent Systems (ICMAS-95)*, pp. 312–319, AAAI, 1995.

[25] M. Bratman, *Shared Agency: A Planning Theory of Acting Together*. New York: Oxford University Press, 2014.

[26] K. Tuyls and G. Weiss, "Multiagent learning: Basics, challenges, and prospects," *AI Magazine*, vol. 33, no. 3, pp. 41–52, 2012.

[27] O. Etzioni, "Intelligence without robots: A reply to brooks," *AI Magazine*, vol. 14, p. 7, Mar. 1993.

[28] V. Singh, G. Singh, and S. Pande, "Emergence, self-organization and collective intelligence – modeling the dynamics of complex collectives in social and organizational settings," in *Computer Modelling and Simulation (UKSim), 2013 UKSim 15th International Conference on*, pp. 182–189, April 2013.

[29] D. C. Dennett, "Precis of the intentional stance," *Behavioral and Brain Sciences*, vol. 11, no. 3, pp. 495–505, 1988.

[30] D. C. Dennett, *The Intentional Stance*. [The MIT Press], 1989.

[31] T. M. Moerland, J. Broekens, and C. M. Jonker, "Emotion in reinforcement learning agents and robots: a survey," *Machine Learning*, vol. 107, pp. 443–480, Feb. 2018.

[32] K. R. Scherer, "What are emotions? and how can they be measured?," *Social Science Information*, vol. 44, no. 4, pp. 695–729, 2005.

[33] R. A. Calvo and S. D'Mello, "Affect detection: An interdisciplinary review of models, methods, and their applications," *IEEE Transactions on Affective Computing*, vol. 1, no. 1, pp. 18–37, 2010.

[34] P. Maes, "How to do the right thing," tech. rep., USA, 1989.

[35] T. Mitchell, *Machine Learning*. McGraw-Hill Education, 1997.

[36] "scikit-learn: machine learning in Python 2014; scikit-learn 1.4.2 documentation — scikit-learn.org." https://scikit-learn.org/stable/, 2024. [Última visita: 15 de noviembre de 2024].

[37] "Diario oficial de la federación."

[38] S. K. . W. M. Jennings, N. R., "A roadmap of agent research and development," *Autonomous agents and multi-agent systems*, vol. 1, pp. 7–38, 1998.

[39] M. Fasli, *Agent technology for e-commerce*. John Wiley & Sons Chichester, 2007.

[40] C. Bourjot, D. Desor, and V. Chevrier, *Stigmergy*, pp. 123–138. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011.

[41] P.-P. Grassé, "Nouvelles expériences sur le Termite de Müller (Macrotermes mülleri) et considérations sur la théorie de la stigmergie," *Insectes Sociaux*, vol. 14, no. 1, p. 73–101, 1967.

[42] M. M. Yezdi Lashkari and P. Maes, "Collaborative interface agents," *BUSCAR*, vol. 1, pp. 7–38, 1234.

[43] N. Gilbert, *Agent-Based Models (Quantitative Applications in the Social Sciences)*. SAGE Publications, Inc, annotated edition ed., Sept. 2007.

[44] W. A. Luna Ramírez, "Modelación basada en agentes (telescopio y microscopio para las humanidades)," Feb. 2016.

[45] P. McCorduck, *Machines Who Think: A Personal Inquiry into the History and Prospects of Artificial Intelligence*. AK Peters Ltd, 2004.

[46] I. Pohl, "John Haugeland. Artificial intelligence: the very idea. Bradford books. The MIT Press, Cambridge, Mass., and London, 1985, ix 289 pp.," *Journal of Symbolic Logic*, vol. 53, no. 2, pp. 659–660, 1988.

[47] B. Margaret, *Artificial Intelligence. Its Nature and Future*. Oxford University Press, 2016.

[48] E. Dussel, *Filosofía de la liberación - Obras selectas XI*. Buenos Aires: Docencia, 1a ed., 2013.

[49] A. Sánchez Vázquez, *Ensayos marxistas sobre filosofía e ideología*. Barcelona: Océano, 1983.

[50] S. Zuboff, *La era del capitalismo de la vigilancia: a lucha por un futuro humano frente a las nuevas fronteras del poder*. Paidós, 2020.

[51] L. Olivé Morett, "Ética aplicada a las ciencias naturales y la tecnología," pp. 181–224, 2003.

[52] M. A. Quintanilla. México: Fondo de Cultura Económica, 2016.

[53] S. Mohamed, M. Png, and W. Isaac, "Decolonial AI: decolonial theory as sociotechnical foresight in artificial intelligence," *CoRR*, vol. abs/2007.04068, 2020.

[54] C. Paizanni, "Enrique dussel - cátedra de pensamiento crítico- sesión 1."

[55] M. Bolom Pale, *A'iel snopel: un ensayo sobre el lenguaje y la filosofía de los pueblos*. Comitán de Domínguez, Centro Regional de Formación Docente e Investigación Educativa, 2020.

[56] M. Taddeo and A. Blanchard, "Accepting moral responsibility for the actions of autonomous weapons systems—a moral gambit," *Philosophy &amp Technology*, vol. 35, Aug. 2022.

[57] A.-K. Oimann, "The responsibility gap and LAWS: a critical mapping of the debate," *Philosophy &amp Technology*, vol. 36, Jan. 2023.

[58] H. Cantrell, "Autonomous weapon systems and the claim-rights of innocents on the battlefield," *AI and Ethics*, vol. 2, pp. 645–653, Nov. 2021.

[59] M. Schreiner, "Ai in war: How artificial intelligence is changing the battlefield," Jan 2023.

[60] D. Allsopp, P. Beautement, M. Kirton, J. Bradshaw, N. Suri, E. Durfee, C. Knoblock, A. Tate, and C. Thompson, "Coalition agents experiment: multiagent cooperation in international coalitions," *IEEE Intelligent Systems*, vol. 17, no. 3, pp. 26–35, 2002.

[61] A. Goldfarb and J. R. Lindsay, "Prediction and judgment: Why artificial intelligence increases the importance of humans in war," *International Security*, vol. 46, pp. 7–50, Feb. 2022.

[62] M. E. O'Hanlon, "The role of ai in future warfare," Mar 2022.

[63] E. d. Gortari, *Indagación de la crítica y de la tecnología*. Grijalbo, 1984.

[64] Y. Huy, *Fragmentar el Futuro. Ensayos sobre la tecnodiversidad*. Caja Negra, 2020.

[65] A. Ng, "Whats next for ai agentic workflows," April 2024.

[66] C. I. M., *Lógica Simbólica*. McGraw Hill, 1998.

[67] A. Tarski, *Introduction to Logic and to the Methodology of Deductive Sciences*. Oxford University Press, 1995.

[68] A. V. Aho, J. E. Hopcroft, and J. D. Ullman, *Estructura de Datos y Algoritmos*. Addison-Wesley Iberoamericana, 1988.

[69] A. Herft, "A teacher's prompt guide to chatgpt. aligned with 'what works best'."

[70] L. Cordina, "Cómo utilizar chatgpt en el aula con perspectiva ética y pensamiento crítico: una proposición para docentes y educadores."

[71] M. Olivera, "Usos de la iag en la universidad." https://www.youtube.com/watch?v=eK84wmdZ5qQ, 2023. [Última visita: 15 de noviembre de 2024].

[72] A. Tiulkanov, "Is it high time to take chatgpt offline?." https://www.linkedin.com/pulse/high-time-take-chatgpt-offline-aleksandr-tiulkanov/. [Última visita: 15 de noviembre de 2024].

[73] A. Tiulkanov, "Is it safe to use chatgpt for your task?." https://www.linkedin.com/posts/tyulkanov_a-simple-algorithm-to-decide-whether-to-use-activity-7021766139605078016-x8Q9. [Última visita: 15 de noviembre de 2024].

Introducción a los Sistemas Multi-Agente Parte I, de Wulfrano Arturo Luna Ramírez, es una obra que se puede encontrar en la página web del repositorio de la UAM Cuajimalpa Concéntric@.

La asistencia editorial estuvo a cargo de Denise Ocaranza.

Casa abierta al tiempo

UNIVERSIDAD AUTÓNOMA METROPOLITANA