Inteligencia Artificial

Curso: 2001/2002
Alumna: Laura M. Castro Souto
Profesores: Vicente Moret Bonillo
Mariano Cabrero Canosa
Eduardo Mosqueira Rey

Índice general

Capítulo 1

Introducción

La **Inteligencia Artificial** (*IA*) es una disciplina relativamente reciente, fruto de trabajos en distintas áreas del pensamiento que fueron definitivamente aglutinados tras el advenimiento de las ciencias de la computación como materia consolidada de investigación y desarrollo.

La **Inteligencia Artificial** está ligada al concepto de *inteligencia*, por cuanto ésta representa una faceta más propia de los seres humanos que de los seres vivos en general, aunque, como veremos, esta última afirmación es discutible y su veracidad dependerá de cómo definamos el concepto de inteligencia.

Podría considerarse que la *IA* es una ciencia que trata de establecer las bases para el posterior desarrollo de un conjunto de técnicas destinadas a dotar a las máquinas de una cierta autonomía. Esta autonomía puede referirse a aspectos muy diversos, como la comunicación con las máquinas en lenguaje natural, la toma de decisiones en un dominio concreto, la toma de decisiones en el tiempo o el aprendizaje.

La IA, como ciencia que nos va a permitir comprender algunos de los principios básicos de la "educación computacional", no hace mucho que ha dejado de ser una disciplina emergente. Su grado de consolidación no es absoluto, sus logros actuales, aunque cualitativamente hablando son importantes, cuantitativamente distan mucho de cubrir los objetivos inicialmente planteados: estamos todavía lejos de poder construir máquinas que piensen, aprendan y creen por sí mismas. No obstante, muchos han sido los esfuerzos empleados en dotar a la IA del peso específico que ya tiene.

La IA, como veremos, maneja *conocimientos*, esto es, *información procesada*; podríamos describirla por tanto como una disciplina informática que se apoya en otras[1] y que se puede abordar desde dos perspectivas: como **ciencia** y como **técnica**.

Ciencia	–	interpreta
Técnica	–	construye
Cultura	–	aglutina

[1] La informática se ocupa del tratamiento y generación de información –datos contextualizados–.

El fin que persigue no es único:

√ simulación (plano físico: construcción)

√ autómatas (plano intelectual: comprensión)

√ robots inteligentes (plano metafísico: poder mágico)

1.1. Cuestiones Preliminares

Es en los textos mitológicos donde aparecen las primeras referencias a los *androides*. Uno de ellos, Talos, obra de Dédalo, es descrito como un gigantesco robot de bronce encargado de la custodia y defensa de la isla de Minos: recorría a diario la su perímetro en busca de visitantes indeseables. Si algún infeliz era descubierto e identificado como enemigo, Talos saltaba dentro de una hoguera, en la que permanecía hasta ponerse incandescente, y luego abrazaba al recién llegado hasta provocarle la muerte.

También en la Mitología encontramos referencias sobre los androides femeninos construidos por Hefaistos, seres de oro macizo dotados de inteligencia (capaces de hablar) y cuya misión era ayudar a caminar a su creador.

Más modernamente, y probablemente fruto de la imaginación popular, encontramos la leyenda del Golem, que puede considerarse como uno de los primeros paradigmas de la IA. El Golem era una estatuilla de arcilla roja a la que un rabino de la comunidad judía de Praga, Dezadel, dio vida a través de un ritual de magia negra. El Golem no hacía nada por iniciativa propia, y su único cometido era el de actual como un fiel esclavo de su creador. Tras una vida pesada y aburrida, se fueron despertando en el Golem instintos que hasta entonces habían permanecido ocultos y, poco a poco, fue liberándose de la tutela de su amo, convirtiéndose en un personaje terrorífico y malvado. Finalmente, Dezadel destruyó al Golem, aunque la leyenda asegura que reaparece en Praga cada treinta y tres años.

Pero quizás los antecedentes históricos más remotos de la IA haya que buscarlos en la antigua Grecia. En concreto, y pertenecientes a la época alejandrina, Arquímedes, Demetrio de Farleria, Architas de Tarente y Herón de Alejandría fueron los verdaderos precursores de una disciplina que hoy se conoce con elnombre de *Automática*.

Arquímedes construyó los mecanismos defensivos que fueron empleados para tratar de contener los ataques de la flota romana sobre Siracusa. Por su parte, Architas de Tarente, más prosaico, construyó una paloma que batía las alas. Demetrio de Farleria dedicó sus esfuerzos a la construcción de un caracol mecánico capaz de avanzar arrastrándose. Finalmente, Herón de Alejandría diseñó y construyó unos actores artificiales capaces de representar escenas de la guerra de Troya. Es en esta época donde podemos buscar los *principios generales de los autómatas*, según los cuales:

- Los mecanismos de un autómata actúan en virtud de su propia estructura interna.
- La acción procede de una adecuada organización de fuerzas motrices, naturales y artificiales.

- La movilidad de los autómatas afecta a todo el conjunto, y no sólo a ciertas partes.

Ya en la Edad Media, encontramos el mayordomo de San Alberto y el león florido de Leonardo, aunque los más reseñables son los trabajos de Llull, que constituyen la primera aproximación a la IA desde el pensamiento. Llull establece en su obra *Ars Magna* las bases de una técnica que aún hoy, con algunas modificaciones, se sigue utilizando en IA.

El *método llulliano* consiste en realizar un *ensayo exhaustivo y sistemático* de todos los procedimientos y principios básicos que pudieran ser útiles para resolver problemas concretos. El método propuesto requiere:

- Adquirir los principios fundamentales conocidos y admitidos por todos en un dominio de aplicación determinado.
- Agotar todas las posibles combinaciones de dichos principios.

La filosofía del método llulliano es clara: al realizar un ensayo exhaustivo de todos los principios fundamentales de un dominio tendríamos que ser capaces de encontrar la solución a nuestro problema, siempre y cuando esté bien definido, y considerando mentes finitas.

No obstante, este método presenta varios inconvenientes:

- Determinar cuáles son realmente los principios básicos de un dominio.
- La inevitable explosión combinatoria.

Tras los trabajos de Llull, y ya en plena Edad Moderna, es obligado mencionar a los Droz, quienes construyeron tres autómatas notables a tamaño natural: uno de ellos fue diseñado para escribir distintos mensajes de hasta cuarenta caracteres; este autómata movía el papel con una mano y la pluma con la otra, mientras los ojos seguían el trazado de la pluma sobre el papel. Otro hacía dibujos variados. El último tocaba el órgano, y presionaba con sus dedos las teclas, mientras su pecho subía y bajaba con el ritmo de su "respiración" y su cabeza acompañaba la música. Estos tres autómatas estaban accionados por complejísimos mecanismos de relojería.

También, por esa misma época, Vaucanson acometía la construcción de los autómatas que, hoy en día, pueden considerarse más evolucionados. En efecto, su flautista, caramillista y el pato que graznaba, batía las alas, comía y hacía la digestión de forma mecánica, son de una perfección notable.

La diferencia fundamental entre estos autómatas y las criaturas de la mitología no es morfológica, sino de actitud: estriba en la manera de comunicarse. Así, para indicarle al escritor de los Droz que debía cambiar el texto, era necesario cambiar su *programa*[2], lo cual requería seis horas de trabajo de un experto relojero. Por el contrario, los autómatas de Hefaistos entendían mensajes hablados.

[2]Programar: instruir, dar indicaciones sobre cómo realizar una tarea.

Históricamente, la aproximación a las ciencias puede realizarse desde ópticas derivadas de la metafísica o desde ópticas más intelectuales. Ambas aproximaciones son válidas y muestran el interés del género humano por explicar fenómenos que aún no se entienden. Pero junto a estos intentos válidos y legítimos aparecen siempre actitudes oportunistas como la que se refleja en la historia del *Malzel Chess Automaton*, un prodigioso autómata que era capaz de jugar al ajedrez al mismo nivel que los mejores especialistas de la época (en realidad era un fraude y quien era un consumado jugador de ajedrez era el enano que se acomodaba en su carcasa).

Curiosamente, Edgar Allan Poe, sabedor de los prodigios que se le atribuían al Malzel Chess Automaton, construyó una prueba lógica según la cual el autómata en cuestión no podía ser auténtico. La argumentación de Poe se basaba en dos pilares fundamentales:

- ▷ Ninguna máquina puede cambiar su estrategia durante un desarrollo pretendidamente intelectual. Las mismas fuentes intelectuales no tiene por qué producir las mismas respuestas ante los mismos estímulos.
- ▷ No hay ninguna máquina que sea capaz de utilizar conocimiento derivado de la experiencia. Los procesos inductivos y de aprendizaje no son propios de ingenios mecánicos.

Lo que Poe no sabía es que su argumentación, que no era más que una traducción del llamado *régimen Lovelace* según el cual una máquina sólo puede hacer lo que se le ordena, es falsa. Hoy sabemos que las máquinas pueden modificar sus estrategias de trabajo durante la resolución de ciertos problemas. Además, incorporan conocimiento derivado de la experiencia (en forma de conocimiento heurístico) y lo utilizan para inferir nuevos hechos y para aprender de su propia experiencia.

Sin embargo, la verdadera historia de la IA se inicia con los deseos de Babbage de que su *máquina analítica* fuese capaz de tratar adecuadamente juegos como el ajedrez. En realidad, lo que Babbage pretendía era construir una máquina capaz de pensar, aprender y crear, y cuya capacidad para realizar estos actos se incrementase hasta que los problemas tratados fuesen del mismo nivel que los destinados a los humanos, proyecto que casi les cuesta la vida a él y a su colaboradora Ada Lovelace.

Pero la curiosidad y el afán de superación son características propias del género humano, y la Historia sigue su curso inexorable. Así, tras la calculadora de Pascal, el sistema binario de Leibniz, la lógica simbólica de Frege, las máquinas lógicas de Stanthome y Sevons y el álgebra de Boole, en 1943 se publican tres artículos teóricos relativos a lo que hoy se conoce como *Cibernética*. En el primero, Wiener, Rosenblueth y Bigelow sugieren distintas formas de conferir fines y propósitos a las máquinas. En el segundo, McCulloch y Pitts ponen de manifiesto de qué modo las máquinas pueden emplear los conceptos de lógica y de abstracción y demuestran que cualquier ley de entrada-salida puede modelizarse a través de una red de neuronas artificiales. En el último de estos artículos, Craik propone que las máquinas empleen modelos y analogías en la resolución de problemas. Mientras, en el MIT se trabajaban modelos que permitiesen establecer un conjunto de principios sencillo que explicasen las actividades de la mente humana, y Ernst desarrollaba su *General Problem Solver*, un planificador.

Mas todas estas ideas no salieron de la pura especulación teórica hasta mediados de los años 50, en los que los ordenadores de la época empezaban ya a ser adecuados para permitir la programación de procesos lo suficientemente complejos[3] (aparece Von Neumann y su arquitectura secuencial de computador).

En 1956, un grupo de investigadores se reúne en el Darmouth College para discutir sobre la posibilidad de construir máquinas genuinamente inteligentes. Entre los investigadores que allí se dieron cita estaban Samuel, que había desarrollado un programa para jugar a las damas, McCarthy, que se dedicaba a la construcción de sistemas en los que llevar a cabo razonamientos de sentido común, y Minsky, que trabajaba sobre un problema de geometría plana con la esperanza de conseguir que el ordenador emplease razonamiento analógico sobre figuras. Junto a éstos, Newell, Shaw y Simon, fueron los verdaderos promotores de la IA, término acuñado en su día por McCarthy.

Tras esta reunión se formaron diversos grupos de científicos que siguieron trabajando de forma independiente. Así, Newell y Simon formaron un equipo con la idea de desarrollar modelos de comportamiento humano, iniciando así la rama *conexionista*[4]. Por su parte, McCarthy y Minsky formaron otro equipo dedicado a la construcción de máquinas inteligentes, sin preocuparse especialmente del comportamiento humano, tal y como continuaría haciendo la rama *simbolista*[5] de la IA. Más tarde surgirían enfoques mixtos[6].

El primer planteamiento supuso la emulación de la actividad cerebral y, en la medida de lo posible, la réplica de su estructura. El segundo, la construcción de sistemas en los que los procedimientos empleados para resolver problemas son de naturaleza tal que, de ser empleados por un ser vivo, éste sería considerado inteligente. Ambas aproximaciones, decididamente divorciadas durante mucho tiempo, son sin embargo necesarias para obtener resultados mínimamente interesantes. La primera porque el estudio y el desarrollo de sistemas inteligentes mediante el uso de simuladores da la oportunidad de alcanzar epistemologías complejas. La segunda por el gran interés que supone el diseño y el análisis de sistemas que sean capaces de resolver problemas intelectualmente difíciles.

Ambos enfoques cubren los objetivos fundamentales de la IA moderna: la comprensión de la inteligencia humana y la utilización de máquinas para adquirir conocimiento y tratar satisfactoriamente problemas complicados. Se asume que el comportamiento inteligente se rige por mecanismos automáticos y parece conveniente potenciar las características diferenciasles de los ordenadores que los hacen mejores que los humanos en algunas tareas.

---

[3] Shanon: *"El ordenador es un simulador de la actividad cerebral"*, *"La información se degrada a medida que es utilizada"* –Teoría de la Información–, *"La información tiende a desordenarse"* –Teoría de la Entropía de la Información–.

[4] Los también llamados *pulcros*; para ellos el ordenador es un campo de pruebas funcional y estructural del cerebro humano.

[5] Los *desaliñados*, que buscan sólo el comportamiento inteligente, manejan conocimiento.

[6] Sistemas *simbólico-conexionistas*, que usan RNA para lo que éstas son buenas –clasificación, reconocimiento de patrones– y simbolismo para controlar esa información.

1.2. Algunas definiciones

Definir es uno de los métodos de descubrir.

Turing *"¿Pueden pensar las máquinas?"*. Test de Turing: la inteligencia pasa a un segundo plano, se centra en la *indistinguibilidad* de comportamientos[7].

Andre Mareaux *"Inteligencia es la posesión de los medios para dominar las cosas o a los hombres"*.

Minsky *"Inteligencia es la forma de resolver problemas que aún no se entienden"*.

Hassentein La inteligencia se define en función de sus propiedades constituyentes:

- no desencadenada
- voluntariedad
- inferencia
- memoria

Es una *descripción fenomenológica*: los seres inteligentes

↪ se comunican

↪ tienen conocimiento interno y autoconocimiento

↪ tienen memoria y son capaces de procesar nuevas experiencias

↪ tienen intencionalidad (lo más difícil)

↪ tienen creatividad

↪ infieren y razonan

Nilson *"La IA es una ciencia que estudia los mecanismos generales necesarios para lograr que los ordenadores hagan cosas que, por el momento, los humanos hacen mejor"*.

Dendral En respuesta a *"¿Es posible diseñar un artefacto que examine unas observaciones y produzca hipótesis relevantes para explicarlas?"*, diseña un sistema que construye la mejor hipótesis para explicar correctamente un conjunto de datos usando mecanismos de razonamiento inductivo y empírico (proceso dirigido por los datos). En el otro extremo, (partiendo de una hipótesis relacionarla con los datos para confirmarla) se tienen los sistemas expertos.

IA Disciplina encargada de diseñar máquinas de forma que éstas sean capaces de llevar a cabo tareas que, de ser realizadas por un ser humano, requerirían algún tipo de inteligencia.

[7] *Tiempo real duro*: respuesta inmediata; *tiempo real blando*: respuesta no inmediata pero más rápida que la del experto humano.

Rama de las ciencias de la computación en la que se intentan encontrar esquemas generales de representación del conocimiento y formalizar procesos de razonamiento coherentes que permitan resolver problemas diferentes en dominios más bien estrechos[8].

Ciencia que utiliza elementos simbólicos y numéricos, conjuntos semánticos, procesos heurísticos, mecanismos lógicos... para emular el comportamiento de los seres humanos.

Son elementos de comportamiento inteligente:

Indexación: sistema endógeno de organización tal que dado un problema sólo el conocimiento relevante es usado.

Curiosidad: necesidad de exploración de posibles soluciones inicialmente no contempladas, planteamiento de nuevas propuestas.

Aprendizaje: incorporación de nueva información potencialmente útil (lo adecuado y en el momento oportuno).

Ampliación, estructuración y armonización: tareas que median entre los conocimientos vacilantes de un novato y el conocimiento sereno de un verdadero experto –empleando heurística–.

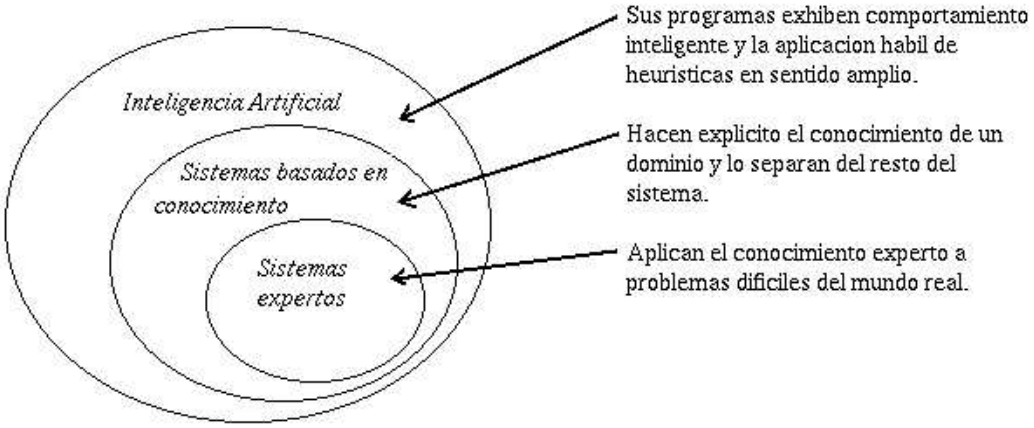


Figura 1.1: Niveles epistemológicos de la IA.

1.3. Consideraciones Generales sobre IA

Los dos planteamientos anteriores dan lugar a una dicotomía difícil de resolver: ¿qué es la IA, una *ciencia* o una *ingeniería*? Como ciencia, debe desarrollar el vocabulario y los conceptos que permiten ayudar a entender y reproducir comportamiento inteligente. Como ingeniería, debe definir y utilizar un conjunto de métodos que permitan adquirir conocimiento de alto nivel, formalizarlo, representarlo según un esquema computacionalmente eficaz, y utilizarlo para resolver problemas en dominios de aplicación concretos.

[8]Esta definición se aproxima mucho a la de *sistema experto*.

La discusión sobre los ámbitos de aplicabilidad de la IA nos permite clasificar (criterio de Waterman) los sistemas inteligentes en tres niveles distintos: *programas de IA*, *sistemas basados en conocimiento* y *sistemas expertos*.

1.3.1. Programas de IA

Los programas de IA exhiben cierto comportamiento inteligente fruto de la aplicación hábil de heurísticas en sentido amplio, entendiendo como *heurística* un tipo de conocimiento difícilmente formalizable, fruto de la experiencia y que se establece implícitamente para tratar de encontrar respuestas más o menos correctas, pero siempre válidas, a un problema concreto. La utilización de conocimiento heurístico no garantiza encontrar soluciones óptimas, pero sí permite garantizar el hallazgo de soluciones aceptables, si existen, a través de los denominados *procesos inferenciales*.

Inferencia es el proceso que permite la comprensión de un significado en función de cierta información relacionada. Su idea está ligada a los procesos de razonamiento, que frecuentemente exigen la realización de varias inferencias para lograr establecer conclusiones válidas. El estudio del *razonamiento* nos permite su clasificación en, al menos, tres *modos* diferentes:

1. Razonamiento deductivo: parte de una premisa general `A1` y de una premisa particular `A2` referida a la primera parte de `A1` y trata de demostrar la premisa particular `A3` referida a la segunda parte de `A1`.

   ```
   A1 : La gripe produce fiebre
   A2 : Luis tiene gripe
   ---------------------------
   A3 : Luis tiene fiebre
   ```

   Este es el modo de razonamiento propio de las matemáticas y su empleo no genera conocimiento nuevo, simplemente aplicamos conocimiento dado sobre situaciones particulares para obtener conclusiones válidas.

2. Razonamiento inductivo o no monótono: parte de dos premisas de naturaleza particular `A2` y `A3` e intenta obtener la aserción general `A1`.

   ```
   A2 : Estos objetos caen
   A3 : Estos objetos tienen masa
   ------------------------------
   A1 : Los objetos con masa caen
   ```

   Este modo de razonamiento está ligado a la experimentación y es propio de ciencias naturales como la física y la química. Genera conocimiento nuevo, basándose en la observación de eventos y en las posibles relaciones que pueden encontrarse. Plantea como problema que puede dar lugar a errores importantes (sustitúyase `con masa` por `son rojos`), que se minimizan mediante un cuidadoso análisis previo de las premisas particulares que se pretenden relacionar. Además, basta un único contraejemplo para refutarlo.

3. Razonamiento abductivo o impreciso: trabaja sobre la plausibilidad de las conclusiones, tratando formalmente de relacionar aserciones de tipo `A1` con aserciones de tipo `A3` para concluir aserciones de tipo `A2`.

```
A1 : Los cuadros de Goya presentan la característica X
A3 : Este cuadro presenta la característica X
-----------------------------------------------------
A2 : Este cuadro es de Goya
```

   Este modo de razonamiento trata de concluir algo sobre la posible relación entre dos aserciones que se sabe que son ciertas, aunque dicha relación puede ser cierta o no. Entre sus características reconocemos que está ligado al concepto de *incertidumbre* (la conclusión viene afectada de incertidumbre, se puede suponer cierta salvo que se demuestre lo contrario), que genera nuevo conocimiento y que es el modo de razonamiento *típico de la IA*, aunque no el único.

1.3.2. Sistemas basados en conocimiento

El siguiente nivel es el de los sistemas basados en conocimiento, en los que los conocimientos del dominio concreto y las estructuras de control que se utilizan para manipularlo se encuentran físicamente separados, lo que va a requerir la definición e implementación de arquitecturas diferentes a las que estamos habituados, y en las que unos y otras puedan ser desarrollados independientemente entre sí, de forma que una misma estructura de control pueda ser utilizada en muchas bases de conocimiento diferentes y viceversa.

1.3.3. Sistemas expertos

Por último, los sistemas expertos pueden considerarse especializaciones de los sistemas basados en conocimiento que utilizan conocimiento particular de un dominio de aplicación concreto para tratar de resolver problemas del mundo real, limitados en tamaño, pero de gran complejidad.

La construcción de sistemas expertos requiere del empleo de técnicas desarrolladas para construir programas de IA y la utilización de las arquitecturas definidas para el desarrollo de sistemas basados en conocimiento, pero además es imprescindible la realización de esfuerzos en aspectos diferenciales como son, por ejemplo, la adquisición del conocimiento y el aprendizaje. La incorporación de información específica de un dominio se simplifica notablemente si se pueden establecer categorías en el conocimiento barajado.

Estas *categorías* pueden definirse en relación al origen y procedencia del conocimiento mencionado, con respecto al experto humano de quien lo extraemos:

- √ *Conocimiento público*, que puede obtenerse directamente a partir de fuentes típicas (manuales, libros), comúnmente aceptado y universalmente reconocido.

- √ *Conocimiento semipúblico*, explícito pero no universalmente reconocido ni comúnmente aceptado, utilizado casi de forma exclusiva por los especialistas del área concreta.
- √ *Conocimiento privado*, no explícito, no universalmente reconocido ni comúnmente aceptado, de marcado carácter heurístico, endógeno de cada uno, fruto de la propia experiencia.

Un sistema de conocimiento pretende familiarizarse con el conocimiento público, implementar el semipúblico y extraer el privado.

Capítulo 2

Resolución de Problemas

No podemos decir que algo o alguien exhibe *comportamiento inteligente* si no explota de manera *eficaz* y *eficiente* un conjunto mínimo de conocimientos. Decimos que un sistema es eficaz cuando es capaz de resolver correctamente un problema, y decimos que es eficiente cuando, además de comportarse eficazmente, optimiza los recursos disponibles. Una arquitectura, natural o artificial, bien definida y estructurada pero vacía, no puede utilizarse para resolver problemas mientras no incorpora procedimientos de resolución y conocimientos propios del dominio de los problemas planteados.

Está claro que no todos los problemas son iguales, por los que los tipos de conocimiento necesarios (las técnicas para abordarlos) van a ser diferentes también. En general, el tipo de problema planteado condiciona la técnica de resolución a emplear. No obstante, la decisión de utilizar una técnica de IA o una técnica convencional de programación puede ser también cuestión de planteamiento. Como norma general, el empleo de técnicas de IA debe permitir la construcción de programas que:

- *Capten generalizaciones*, de forma que cada situación individual que se produzca no tenga que ser representada de forma separada, sino que todas aquellas situaciones que compartan propiedades deben ser agrupadas.
- *Hagan explícito su conocimiento* (que generen explicaciones en lenguaje natural), al objeto de facilitar su comprensión.
- *Puedan actualizarse continuamente*, de forma que sea factible modificar el conocimiento sin tener que manipular ni alterar todo el programa.
- *Puedan ser empleados en muchas situaciones*, aún cuando las respuestas que generen sean parcialmente correctas o imprecisas.

La aplicación de estos criterios en la resolución de problemas de dominios concretos, conduce a la construcción de programas con características esencialmente diferentes a los construidos mediante técnicas convencionales:

1. Los programs de IA deben ser empleados para tratar fundamentalmente dominios simbólicos, mientras que los programas convencionales son particularmente idóneos para tratar dominios numéricos.

2. La búsqueda de soluciones en los dominios apetecidos por la IA se realiza a través de procesos heurísticos[1] en los que los pasos hacia la solución suelen ir implícitos. Por el contrario, los programas convencionales emplean procedimientos algorítmicos de búsqueda, con pasos explícitos hacia la solución.
3. En los programas convenicionales la información y el control se encuentran físicamente integrados en una misma estructura, mientras que en los programas de IA el conocimiento del dominio y las estructuras de control suelen estar físicamente separados, dando como resultado arquitecturas mucho más modulares.

Como siempre ocurre, la diferencia no es lo suficientemente evidente como para permitirnos establecer una frontera clara entre los "problemas de la IA" (aquéllos para cuya resolución está indicada la aplicación de técnicas de IA) y los "problemas convencionales" (aquéllos para cuya resolución están indicadas las técnicas de programación convencional). Sin embargo, un análisis cuidadoso del dominio nos puede dar pistas a la hora de elegir una u otra filosofía, y evitar así errores provocados por el hecho de no ajustar la técnica al tipo de problema que queremos resolver.

Supongamos que podemos definir a priori un conjunto de situaciones posibles y unas "reglas del juego" potencialmente útiles, y también que podemos diseñar unas estrategias generales, que investiguen la aplicabilidad de las reglas del juego sobre los estados para obtener otros estados, de forma que sea el ordenador quien encuentre por sí mismo la solución al problema. Nótese que este enfoque es de naturaleza no determinística, y que un programa que siguiese este planteamiento sería muy general, podría utilizarse para resolver un número ilimitado de problemas, ya que sólo cambiarían las reglas del juego aplicables, en función del estado del problema en cada momento. Para conseguir esto, en IA se define el *espacio de estados* del problema.

2.1. Espacio de Estados

En IA es útil definir el dominio del problema que queremos resolver como un **espacio de estados**, que es una descripción formal del universo de discurso y está constituido por los siguientes elementos:

- Un conjunto de estados iniciales I.
- Un conjunto de operadores O que definen operaciones permitidas entre estados. En el espacio de estados no todas las operaciones definidas van a ser aplicables siempre: en un momento dado, el conjunto de estados que representa la situación actual de nuestro problema es quien determina el subconjunto de "operadores aplicables" (o "relevantes") del conjunto

[1] El conocimiento heurístico es un conocimiento difícilmente formalizable, fruto de la experiencia y que se establece implícitamente para tratar de encontrar respuestas más o menos correctas, pero siempre válidas, a problemas de dominios concretos.

de operadores previamente definido. Para que un operador pueda ser considerado relevante debe cumplir un conjunto de requisitos mínimos.

- ◦ Un conjunto de metas M (objetivos) que cumplen los requisitos suficientes para ser consideradas soluciones aceptables de nuestro problema[2]. La solución no es la meta, sino la trayectoria de búsqueda, la secuencia de estados que conduce del inicial al final.

Llamaremos *nodo* a un punto concreto del *espacio de estados* y *estado* a la secuencia de pasos que se ha seguiro para llegar a un nodo.

El *espacio de estados* (I, O, M) es útil porque permite describir formalmente un problema como un conjunto de transformaciones desde unas situaciones dadas hasta unas situaciones deseadas, a través de un conjunto de operaciones permitidas. Más concretamente, nos permite contemplar el proceso global de solución de un problema como:

- ▷ la aplicación de un conjunto de técnicas conocidas, cada una de ellas definida como un paso simple en el espacio de estados, y
- ▷ un proceso de búsqueda o estrategia general de exploración del espacio de estados.

Formalmente:

Si $I = [i_1, i_1, \ldots, i_n]$ define al conjunto de estados iniciales, $O = [o_1, o_2, \ldots, o_m]$ define al conjunto de operadores potencialmente útiles y $M = [m_1, m_2, \ldots, m_t]$ define al conjunto de metas o estados finales, la *búsqueda* se define como el proceso de exploración del espacio de estados que produce $O : (I \rightarrow M)$, una evolución desde los estados iniciales hasta los estados finales obtenida tras la aplicación de un conjunto de operadores. En este mismo contexto, una transición simple en el espacio de estados puede representarse

$$o_x : (i_z \rightarrow i_w)$$

con $i_z, i_w \in I$, $i_z, i_w \notin M$ y $o_x \in O$.

Si $i_w \in M$, entonces i_w representa una *solución aceptable* para nuestro problema. La llamada *prueba de meta* o *test de realización* aplicada a la descripción de un estado permitirá decidir si alguno de los nuevos estados generados se trata de un estado meta. Cualquier otro estado alcanzado durante la búsqueda que no pertenezca al conjunto de metas puede ser considerado como un nuevo estado inicial del problema.

Este formalismo de resolución normalmente se traduce en la creación de programas menos eficientes que los convencionales, pero mucho más flexibles y generales. En cualquier caso, la resolución de problemas en IA requiere siempre una descripción formal y manejable del problema, esto es, la elaboración de un *modelo computacional* del universo

[2]En IA se habla de *soluciones aceptables*, no de *mejor solución*.

de discurso o dominio del problema.

El *espacio de estados*, aunque imprescindible para la representación formal de un problema de IA, tan sólo nos proporciona lo que podríamos llamar el "soporte físico" del dominio. El aspecto más dinámico de obtención de soluciones se materializa a través de los llamados *procesos de búsqueda*, mecanismos generales de exploración del E.E.

El concepto de *búsqueda* está íntimamente ligado a la aplicación de operadores relevantes. Más generalmente, la búsqueda se puede asimilar a los procesos de control qe guían la ejecución de un programa de IA, según los cuales el sistema debe ser capaz de decidir en todo momento cuál será su próximo movimiento en el espacio de estados. Conviene que la búsqueda sea *sistemática* (para evitar "rodeos" innecesarios) y que la aplicación de cada operador provoque un movimiento que genere la aparición de un estado nuevo (que no haya sido generado ya, para evitar hacer y deshacer continuamente). Así, nos encontramos con diversas estrategias:

Estrategia 1 A priori, una buena posibilidad podría ser la de aplicar el primer operador, según un orden establecido, que cumpla los requisitos del estado actual, y verificar si el nuevo estado es meta. Esta estrategia de búsqueda es sistemática, pero no observa la condición de generar estados nuevos, con lo cual puede caer en un bucle. Además, hay operadores cuya aplicación se repite, lo cual no es deseable tampoco.

Estrategia 2 Intentando evitar la aplicación repetitiva de operadores, pero manteniendo el criterio de sistematicidad, podría seguirse este otro esquema:

1. seleccionar los operadores que verifiquen las precondiciones del estado actual
2. descartar aquéllos que ya hayan sido aplicados
3. aplicar el primero de los restantes[3].
4. test de realización para el nuevo estado

Nótese que esta estrategia nos obliga a definir estructuras adicionales que nos permitan comprobar si un operador ha sido aplicado ya o no; además, aún no evita la generación de estados ya generados, y puede no encontrar solución.

Estrategia 3 Tratando de impedir de forma explícita los inconvenientes señalados en la primera estrategia, podría definirse el esquema siguiente:

1. seleccionar los operadores que verifiquen las precondiciones del estado actual
2. descartar los que ya hayan sido aplicados
3. descartar aquéllos cuya aplicación no genere un estado nuevo
4. aplicar el primero de los restantes
5. prueba de meta

[3]Si en este punto los operadores se ordenasen aleatoriamente podríamos llegar a una solución "por casualidad" o conseguir diferentes soluciones para los mismos estados inicial y final.

Nuevamente precisaríamos definir estructuras adicionales (aparte de saber qué operadores hemos aplicado ya, debemos comprobar si un estado es nuevo o no), y podemos aún no encontrar solución. Parece evidente que hemos de sacrificar algo si queremos encontrarla.

Estrategia 4 Seguiría los pasos:

1. selección de los operadores que verifiquen las precondiciones del estado actual
2. descarte de los que no generen estados nuevos
3. aplicación del primero de los restantes
4. test de realización

Ahora sí se encuentra una solución, pero parece claro que no es una forma óptima de proceder.

El motivo por el cual se ha ilustrado de este modo el hipotético proceso que el sistema ha seguido para encontrar una solución es tratar de evidenciar las diferencis existentes entre los programas convencionales y los programas de IA. El conocimiento del sistema (operadores, precondiciones) se define de manera totalmente independiente de la forma de utilizarlo (estrategias, mecanismos de control del conocimiento). Además, a medida que vamos explorando diversas estrategias de resolución, vamos encontrando la necesidad de definir estructuras auxiliares. Y, por último, la solución encontrada por el sistema es "aceptable" (cumple los requisitos para ser meta), pero puede no ser la mejor. La validez de la solución encontrada dependerá mucho del tipo de problema planteado.

Las distintas estrategias de búsqueda que se presentan en este capítulo tratarán de resolver este tipo de problemas.

2.2. Características Generales de los Procesos de Búsqueda

Como ya hemos mencionado, los programas de IA deben ser flexibles y generales, de forma que, independientemente del universo de discurso, permitan la utilización de técnicas aplicables a la resolución de cualquier problema, y que sean de una eficiencia, por lo menos, aceptable. Surgen así las llamadas **técnicas de búsqueda de propósito general**, conocidas también como **métodos débiles de exploración del E.E.** Estas técnicas derivan de la idea de búsqueda heurística, y pueden definirse independientemente de cualquier tarea particular, o del dominio concreto considerado[4].

Cada técnica de búsqueda tiene sus ventajas, inconvenientes e idiosincrasia particular. Antes de decidirnos por una u otra, es conveniente estudiar su idoneidad en relación al tipo de problema y dominio planteado. Esta idoneidad puede establecerse analizando un conjunto de características esenciales que condicionan el proceso de búsqueda:

[4]Evidentemente, esta afirmación no debe tomarse al pie de la letra: el tipo de tarea a resolver y el dominio del problema influyen claramente en la elección de una u otra técnica; aunque conceptualmente todas son igualmente aplicables, unas son más apropiadas que otras.

- √ dirección del proceso de búsqueda
- √ topología del proceso de búsqueda
- √ representación de los estados por los que discurre la resolución del problema
- √ criterios establecidos y procedimiento definido para la selección sistemática de los operadores relevantes en función de los estados alcanzados
- √ posibilidad de optimizar los procesos de búsqueda mediante el empleo de *funciones heurísticas*[5] (usar conocimientos sobre la propia búsqueda)

2.2.1. Dirección del proceso de búsqueda

Existen dos direcciones fundamentales que podemos definir a la hora de configurar un proceso de búsqueda determinado:

- ▶ desde los estados iniciales hacia los estados meta, mediante la generación de estados intermedios obtenidos tras la aplicación sucesiva de operadores relevantes (postura: *¿qué hago con lo que tengo?*):

$$EE(I, O, M) \qquad \Rightarrow \qquad O : (I \rightarrow M)$$

- ▶ desde los estados meta hacia los estados iniciales, investigando qué estados previos al estado (o estados) meta, y qué operadores aplicables, nos producen una transición deseada (postura: *¿qué tendría que haber hecho para llegar aquí?*):

$$EE(I, O, M) \qquad \Rightarrow \qquad O : (M \leftarrow I)$$

La primera dirección definida configura un **razonamiento progresivo** o **dirigido por los datos**, mientras que la segunda configura un **razonamiento regresivo** o **dirigido por los objetivos**.

Entre ambas situaciones límite, encontramos cierto tipo de problemas para los cuales es conveniente emplear *estrategias mixtas* de búsqueda. Así, en algunos casos es conveniente iniciar un proceso dirigido por los datos y, llegado un punto, cambiar la dirección de la búsqueda, o viceversa, iniciar un proceso dirigido por los objetivos, que permita el establecimiento de un conjunto de hipótesis razonables, y luego confrontar las hipótesis con los datos, a través de un proceso progresivo.

Sea como fuere, parece claro que la dirección del proceso condiciona, al menos en parte, los resultados y la eficiencia del sistema, pero... ¿cuándo es aconsejable optar por una u otra? ¿Existe algún criterio que nos permita discriminar entre ambas opciones?

La elección sobre la dirección de búsqueda más conveniente debe considerar tres aspectos diferentes:

[5] La mejor *función heurística* nos daría, idealmente, la mejor solución en tiempo 0.

- ⋆ tamaño relativo de los conjuntos I y M (es preferible explorar el espacio de estados de forma que progresemos desde un conjunto inicialmente pequeño de información de partida hacia un conjunto mayor de estados)
- ⋆ factor de ramificación (número promedio –estimación– de estados que se pueden alcanzar desde uno dado, número de operadores aplicables llegados a un momento dado –profundidad–; influye en la eficacia del proceso de búsqueda, de modo que trataremos de explorar el espacio de estados según la dirección del menor factor de ramificación[6])
- ⋆ inclusión de estructuras explicativas como requisito inicial en el diseño de nuestro sistema inteligente (si el programa debe ser capaz de "explicar" su proceso de razonamiento, es conveniente que lo realice en la dirección que concuerde más aproximadamente con la forma de razonar del usuario humano)

2.2.2. Topología del proceso de búsqueda

Una forma sencilla de explorar el espacio de estados es generar *dinámicamente*[7] un árbol, partiendo de un determinado estado, inicial o final, y expandirlo tras la ejecución de uno o varios operadores relevantes[8].

Es totalmente viable que un mismo estado sea generado durante la exploración de diversos caminos, lo que supone un esfuerzo adicional de computación que se traduce en una menor eficiencia del proceso de búsqueda. Este inconveniente puede mitigarse cambiando la topología del proceso y convirtiendo el árbol en un grafo dirigido.

En la tabla 2.1 (página 24) se expone un mecanismo de conversión completo de árbol de búsqueda a grafo de búsqueda, ya que, además de cambiar la topología del proceso, registra el mejor camino cada vez que se genera un nuevo estado.

La utilización de grafos de búsqueda reduce los esfuerzos de exploración del espacio de estados, aunque tiene el inconveniente de que obliga a comprobar si cada "nuevo" estado generado pertenece ya al conjunto de estasod generados en pasos anteriores.

Las topologías en árbol suelen causar problemas de memoria, pero la búsqueda puede ser más rápida. Las topologías en grafo, aparte de ser conceptualmente más correctas, minimizan los problemas de memoria, aunque pueden hacer disminuir la eficiencia al tener que efectuar comprobaciones frecuentes. Así pues, ambos esquemas tienen ventajas e inconvenientes, de modo que en último término la elección depende del dominio del problema planteado.

[6] Aunque, por desgraca, suele ir normalmente en la dirección contraria a la que indica el primer punto (tamaño relativo de I y M).

[7] En IA los árboles de decisión son siempre implícitos, sólo se materializan cuando se ejecuta un determinado proceso inferencial.

[8] El número de operadores relevantes ejecutados depende de la técnica de exploración elegida, como veremos.

```
(a) Generar (uno o más) estados
      tras la aplicación de (uno o más) operadores relevantes
(b) Examinar el conjunto de estados generados;
     para cada uno:
      (b.1) Si el estado es nuevo
              añadirlo y volver a (a)
      (b.2) Si ya existía
              descartarlo e ir a (c)
(c) Añadir un enlace entre el nodo que se está expandiendo
      y su sucesor
(d) Recorrer el nuevo camino desde el principio
      (d.1) Si es más corto
              insertarlo como mejor camino y
                propagar el cambio
                reorganizar el grafo si es necesario
                volver a (a)
      (d.2) Si no es más corto
              volver a (a)
```

Cuadro 2.1: Algoritmo de transformación de árbol a grafo.

2.2.3. El problema de la representación

De acuerdo con las estructuras definidas en el espacio de estados, el problema de la *representación* puede estudiarse desde tres perspectivas diferentes:

- Representación de los objetos, entidades relevantes o hechos del dominio (estructura del conocimiento) –naturaleza estática–.
- Representación de las relaciones entre objetos, entidades relevantes o hechos del dominio (estructuras que nos permiten transitar por el espacio de estados: operadores) –naturaleza dinámica–.
- Representación de las secuencias de estados surgidas durante los procesos de búsqueda (de la estrategia y los mecanismos de control necesarios para organizar la búsqueda convenientemente, representación dinámica del proceso de búsqueda de soluciones).

La representación de los nodos y la representación de las relaciones entre los nodos definen el problema de la *representación del conocimiento*, que veremos más adelante.

Aunque de las tres perspectivas la más relacionada con la búsqueda es la representación de las secuencias de estados, todas ellas están estrechamente relacionadas y, normalmente, la elección de determinados esquemas de representación para entidades y relaciones suele condicionar el esquema de representación idóneo para las secuencias de estados.

2.2.4. Selección sistemática de operadores relevantes

Ya hemos comentado cómo la aplicación de un determinado operador sobre un estado dado produce un nuevo estado, pero... ¿cómo podemos reconocer, de entre el conjunto global de operadores potencialmente útiles, aquéllos que realmente son aplicables a nuestro estado actual? ¿Cómo podemos extraer del conjunto global de operadores un subconjunto de *operadores relevantes*?

Este problema define lo que en IA se denomina *emparejamiento*, que constituye una de las tareas más costosas y lentas de los programas de IA. Existen varios tipos de *emparejamiento*[9], cada uno con sus ventajas e inconvenientes, y la elección del tipo de emparejamiento suele depender del esquema utilizado para representar el conocimiento:

- El *emparejamiento literal* realiza una búsqueda simple a través de todos los operadores del conjunto O, analizando las precondiciones de cada uno en el contexto del estado actual considerado y extrayendo los que las verifiquen.

  Este tipo de emparejamiento presenta el problema de que su eficiencia está muy ligada al número de operadores definidos en el conjunto O; si consideramos que los problemas realmente interesantes en IA requieren la utilización de gran número de operadores, deducimos que este método suele ser intrínsecamente ineficiente en los problemas reales. Por otra parte, para un estado particular no siempre es evidente que un operador determinado sea aplicable, a veces es difícil saber si un estado satisface o no las precondiciones del operador (puede ocurrir que el operador empareje completamente con el estado, o que sus precondiciones sean un subconjunto de la descripción del estado, o que coincidan sólo parcialmente, o bien que no coincidan en absoluto), de modo que sólo es útil en dominios pequeños en los que la exploración de patrones, costosa y lenta, se vea compensada por lo restringido del conocimiento involucrado.

  Como ventaja, es un método muy informativo.

- El *emparejamiento con variables* es de naturaleza no literal, y es especialmente útil cuando el problema que tratamos de resolver requiere una búsqueda extensa en la que no haya variables involucradas. Es más descriptivo que el anterior, y reduce el número de operadores. Lo ilustraremos con un ejemplo.

  Tengamos los siguientes *hechos del dominio* (describen la situación actual del problema):

  ```
  HIJO (María, Juan)
  HIJO (Juan, Pedro)
  HIJO (Pedro, Tomás)
  HIJA (Pedro, Rosa)
  HIJA (Juan, Ana)
  HIJA (Ana, Rosa)
  ```

[9]Proceso de selección de operadores relevantes.

donde `HIJO (A,B)` significa que `B` es hijo de `A`, y los *operadores*:

```
op1 : HIJO (x,y) AND HIJO (y,z) -> NIETO (x,z)
op2 : HIJA (x,y) AND HIJO (y,z) -> NIETO (x,z)
op3 : HIJO (x,y) AND HIJA (y,z) -> NIETA (x,z)
```

Encontrar un estado meta supone encontrar un estado que incluya los mismos hechos que el estado de partida y, además, algún hecho que indique que realmente es meta (*hecho inferido*). Si quisiéramos saber "quién es nieto de Juan" nos interesaría aplicar los operadores 1 ó 2, puesto que son los que concluyen sobre la existencia de un nieto. La meta se obtendría inmediatamente con sólo sustituir `x` por Juan, pero para ello tendríamos que encontrar primero un `y` que verificase `HIJO (Juan,y) AND HIJO (y,z)` para algún valor de `z`. Esto supondría comprobar todos los hijos de Juan y verificar que alguno de ellos tenga a su vez un hijo, o comprobar entre todos los que tengan algún hijo que alguno de ellos es a su vez hijo de Juan. Frecuentemente, encontraremos muchos valores que satisfagan los predicados por separado, pero muy pocos que los satisfagan todos.

Una vez identificados los operadores aplicables en el estado actual, y en el caso de que hayamos podido identificar más de uno, es cuando se presenta el conflicto: debemos ser capaces de poder elegir de entre dicho conjunto de conflicto la utilización del operador que "a priori" nos ofrezca más garantías de éxito en la búsqueda de un camino adecuado hacia la meta. Un enfoque correcto de este problema de *resolución de conflictos* es fundamental en la construcción de sistemas inteligentes.

Aunque muchas veces los mecanismos idóneos de resolución de conflictos son dependientes del universo de discurso, del esquema de representación elegido y del procedimiento de búsqueda empleado, también es cierto que toda estrategia de resolución de conflictos debe respetar los siguientes criterios generales:

- si puede evitarse, no deben aplicarse operadores que ya hayan sido utilizados
- deben tratar de aplicarse primero operadores que emparejen con los hechos más recientemente incorporados a la base de hechos que describe nuestro estado actual, de manera que utilicemos información más real y actual (lo que obliga a ser capaces de identificar la secuencia temporal en la incorporación de hechos; eficiencia inferencial y computacional)
- se tratará de aplicar primero operadores con precondiciones más restrictivas, más específicos (buscando mayor discriminación, acotar el problema)
- de no darse ninguna de las condiciones anteriores, se seleccionará uno (o varios) operadores al azar

2.2.5. Funciones heurísticas

Se conoce como *función heurística* a aquella función de carácter numérico que permite cuantificar el beneficio de una transición efectuada en el espacio de estados del dominio del problema a resolver. Suelen resultar útiles a la hora de optimizar los procesos de búsqueda, ya que intentan guiar la exploración del espacio de estados en la dirección más provechosa, sugiriendo el *mejor* camino a seguir (a priori) cuando disponemos de varias alternativas, intentando llegar a la solución de manera intuitiva (nos dice cuán lejos estamos de llegar a la solución).

2.3. Estrategias de Exploración del E.E.

La eficacia del proceso de búsqueda viene frecuentemente determinada por la estrategia empleada y por los mecanismos diseñados para controlar la aplicación del conocimiento del dominio.

Está claro que, para cada universo de discurso, siempre podremos establecer estrategias específicas (técnicas ad-hoc, concretas, para casos concretos) de exploración del espacio de estados, es decir, **técnicas de búsqueda de propósito específico**. No obstante, desde la perspectiva de la IA, son mucho más interesantes las estrategias genéricas que unen a su generalidad una eficiencia razonable. Tales estrategias genéricas se agrupan en lo que habitualmente se denominan **métodos débiles de exploración del espacio de estados**[10].

Cualquier método débil de exploración del E.E. configura una búsqueda que será de uno de los siguientes tipos:

- En anchura.
- En profundidad.
- Mixta profundidad-anchura.

2.3.1. Búsqueda preferente por amplitud (anchura)

La *búsqueda preferente por amplitud* o *búsqueda en anchura* trata de generar amplios y crecientes segmentos en el E.E. y en cada nuevo nivel generado verifica si el objetivo ha sido alcanzado antes de pasar al siguiente. Su característica fundamental es, pues, que se expanden todos los nodos de un mismo nivel antes de acceder a nodos de niveles inferiores[11].

La implementación computacional del método utiliza dos listas de nodos:

↪ `ABIERTOS`, nodos que se han generado y a los que se les ha aplicado la función de evaluación pero que aún no han sido examinados (no se han generado sus sucesores)

[10] Se denominan así porque el conocimiento que añaden al problema es *débil*.

[11] Puede verse, pues, como una materialización del *ars magna* de Ramón Llull.

↪ `CERRADOS`, nodos que ya han sido examinados (esta lista tiene la finalidad de evitar la creación de ciclos en el proceso de búsqueda)

Cada nodo n del árbol de búsqueda mantendrá enlaces a todos sus sucesores, pero sólo a un único predecesor. El algoritmo se muestra en la tabla 2.2 (página 28).

```
1. Colocar el nodo inicial en ABIERTOS
2. Aplicar el test de realización a este nodo;
   Si es meta,
     salir e informar de la solución
3. Si ABIERTOS está vacía
     salir e informar del fallo
4. Obtener el primer nodo (N) de ABIERTOS y
   añadirlo a CERRADOS
5. Expandir N generando todos sus sucesores aplicando
   todos los operadores relevantes;
   Si no hay sucesores,
     volver a 3
   sino
     añadir los sucesores al final de ABIERTOS
     y actualizar sus enlaces para que apunten a N
6. Aplicar a los sucesores el test de realización;
   Si alguno de ellos es meta,
     salir e informar de la solución
     siguiendo los enlaces hasta el nodo inicial
7. Volver a 3
```

Cuadro 2.2: Algoritmo de Búsqueda en Anchura.

Los caminos que se exploran paralelamente y no resultan en solución, se dicen *no resolutivos*; nada indica que no puedan ser parte de una solución a otro nivel.

Los procedimientos de búsqueda en anchura, por ser exhaustivos (y sistemáticos, ya que se aplican todos los operadores que pueden emparejarse con todos y cada uno de los nodos del nivel considerado), nos permiten asegurar que, en espacios de estados finitos y bien construidos, el sistema *siempre encontrará la solución* al problema, y además, dicha solución será la *mejor*.

Desgraciadamente, los métodos en anchura son impracticables en dominios amplios, ya que el número de nodos generado en cada nivel sucesivo crece de manera exponencial y, consecuentemente, las necesidades de memoria y el tiempo computacional empleado en la búsqueda también lo hacen.

2.3.2. Búsqueda preferente por profundidad

A diferencia de los métodos en anchura, los métodos en profundidad seleccionan un camino determinado y siguen por él hasta agotarlo completamente. En los métodos en profundidad *puros* el test de realización se efectúa cada vez que se genera un nuevo nodo. Puede ocurrir que el camino recorrido sea resolutivo y lleve a una solución del problema o puede que agotemos todas las posibilidades de expansión sin haber encontrado nada. En este último caso hay que efectuar una *vuelta atrás* y explorar otro camino diferente (ver algoritmo[12] en tabla 2.3, página 29).

```
1. Colocar el nodo inicial N en CAMINO
2. Aplicar el test de realización a este nodo.
   Si es meta,
     salir
3. Expandir N aplicando el primer operador no aplicado
   para gener un sucesor S.
4. Si ningún operador es aplicable,
     salir
5. En caso contrario,
     realizar una búsqueda en profundidad
     partiendo de S
```

Cuadro 2.3: Algoritmo de Búsqueda en Profundidad.

Los métodos en profundidad son sensibles a la "posición relativa" de los operadores en una lista, ya que este es el criterio que se utiliza para la selección de operadores en caso de conflicto. Desde una perspectiva computacional, requieren menos recursos de memoria que los métodos en anchura, ya que consideran un espacio de búsqueda más limitado, pero pueden tratar de explorar caminos muy largos habiendo alternativas mejores e incluso no llegar nunca a alcanzar una solución; no está clara su sistematicidad (la búsqueda en profundidad "pura" es un procedimiento sistemático, pero permite que encontremos la solución "por casualidad"[13], sin que tenga, por supuesto, que ser la mejor) y hay que hacer un montón de comprobaciones, por lo que se es más ineficiente desde el punto de vista temporal.

En ocasiones, los caminos generados por los métodos en profundidad no se exploran completamente, abandonándose la búsqueda tras alcanzarse sin éxito una determinada profundidad. Ello puede suponer que consideremos que un determinado camino es infructuoso cuando, en realidad, se estaba muy cerca de la solución.

Lo normal, en la práctica, es emplear **estrategias mixtas** y desarrollar procedimientos que combinen, de una forma u otra, características de los métodos en anchura y profundidad. A continuación veremos varios de estos métodos.

[12] La ruta con la solución en este algoritmo queda almacenada en la lista de nodos `CAMINO`.

[13] Este aspecto se enfatiza si, tras cada expansión, los operadores aplicables se reordenan aleatoriamente.

2.3.3. Generación y prueba

El método de *generación y prueba* es un procedimiento de búsqueda *en profundidad* casi puro, en el que deben recorrerse caminos completos antes de realizar ninguna comprobación (lo que cambia, pues, es sólo el lugar donde se realiza la prueba de meta, que se limita a responder afirmativa o negativamente acerca de la validez del camino generado). En su forma más sistemática, el método de generación y prueba es una búsqueda exhaustiva en el espacio del problema (ver tabla 2.4, página 30).

```
1. Generar una solución posible
2. Aplicarle la prueba de meta
3. Si es solución,
     parar
   sino
     volver a 1
```

Cuadro 2.4: Algoritmo de Búsqueda Mixta *Generación y Prueba*.

Si la generación es sistemática y exhaustiva, siempre encontraremos la solución, si existe. Sin embargo, si el dominio de prueba es muy amplio, la exploración del espacio de estados mediante este método puede requerir demasiado tiempo.

Los procedimientos de búsqueda vistos hasta ahora tienen una característica común: forman parte de las denominadas *búsquedas ciegas*. Parte de los problemas que aparecen en ellos pueden ser resueltos empleando las denominadas *técnicas heurísticas* de búsqueda, que suelen mejorar la eficiencia de los procesos de resolución de problemas sacrificando la exhaustividad de la respuesta, dejando de considerar algunos caminos que parece improbable que conduzcan a la solución.

Las *estrategias de búsqueda informada* tratan, pues, de optimizar los procesos de búsqueda utilizando funciones heurísticas que les permitan guiar la exploración del E.E. en la dirección más provechosa, sugiriendo el "mejor" camino a seguir cuando se dispone de varias alternativas.

2.3.4. Ascensión a colinas

El procedimiento de *ascensión a colinas* es una variante del método de profundidad en el que la selección del siguiente nodo a expandir se realiza de acuerdo con alguna medición heurística que permite estimar la distancia que queda por recorrer hasta la meta. En su forma más sencilla, el algoritmo de ascensión a colinas se presenta en la tabla 2.5 (página 31).

A diferencia de la búsqueda en profundidad, con este algoritmo una rama no tiene por qué ser explorada hasta agotarse, sino que el proceso de expansión terminará en el momento en que se encuentra un nodo sucesor que no mejora el estado actual.

```
1. Colocar el nodo inicial N en CAMINO
2. Aplicar el test de realización al nodo N;
   Si es meta,
     salir
3. Repetir hasta encontrar un sucesor
     3.1. Aplicar a N un operador no aplicado
          para generar un sucesor S
     3.2. Si no hay ningún operador aplicable,
            salir
     3.3. En caso contrario,
            aplicar la función heurística a S
     3.4. Si mejora la función heurística del estado actual,
            ir a 4
     3.5. Si no,
            volver a 3.1
4. Continuar el proceso de búsqueda partiendo de S
```

Cuadro 2.5: Algoritmo de Búsqueda *Ascensión a colinas*.

También se podría decir que el método de ascensión a colinas es similar a generación y prueba, pero cuando se llega a una situación en la que hay que hacer *backtracking* se aplican varios operadores en el nodo predecesor, es decir, se abre una pequeña componente en anchura y se comprueba si alguno de los nodos generados es meta; si no lo es ninguno, se aplica una función heurística para continuar por alguno de ellos.

Una variante útil del método de ascensión a colinas simple consiste en considerar todos los posibles movimientos a partir del estado actual y elegir el mejor de ellos como nuevo estado. Este método se denomina **ascensión por la máxima pendiente** o **búsqueda del gradiente** (ver tabla 2.6, página 32) y, contrasta con el método básico, en el que el primer estado que parezca mejor que el actual se selecciona como siguiente.

Tanto la ascensión a colinas básica como la de máxima pendiente pueden no encontrar una solución si caen en un estado del que no es posible generar nuevos estados mejores que él, pero que aún no es solución. Esto ocurre cuando el proceso de búsqueda se encuentra con un *máximo local*, una *meseta* o una *cresta*. Considerando el E.E. como una superficie n-dimensional en la que cada punto está definido por los valores que definen ese estado y su valor de la función heurística:

- ▷ Un ***máximo local*** es un estado puntual mejor que cualquiera de sus vecinos pero peor que otros estados más lejanos[14]. Cuando están próximos a la solución, los *máximos locales* se denominan *estribaciones*.
- ▷ Una ***meseta*** es un área plana del espacio de estados en la que toos

[14]Llegados a uno, cualquier operación o movimiento que hagamos nos llevará a un estado aparentemente peor, aunque en realidad no estemos aproximando a la solución.

```
1. Colocar el nodo inicial N en el CAMINO
2. Aplicar el test de realización a este nodo;
   Si es meta,
     salir
3. Aplicar a N todos los operadores disponibles y
   generar todos sus sucesores Sj.
4. Si no hay ningún operador aplicable,
     salir
5. En caso contrario,
     aplicar la función heurística a todos los Sj
6. Si la función heurística de algún Sj mejora la del estado actual,
     llamar a Sj MEJORNODO e ir a 8
7. En caso contrario,
     salir
8. Continuar el proceso de búsqueda partiendo de MEJORNODO
```

Cuadro 2.6: Algoritmo de Búsqueda *Ascensión por máxima pendiente.*

los estados individuales tienen un mismo valor de la función heurística, de suerte que no es posible determinar cuál es la mejor dirección para continuar la búsqueda.

- ▷ Una ***cresta*** es un tipo especial de *máximo local*, un área del espacio de estados que tiene estados con mejores valores de la función heurística que los de regiones colindantes, y además posee una inclinación, pero la orientación de esta región alta hace que sea imposible atravesarla mediante transiciones simples.

Estos inconvenientes del método tienen difícil solución. Como norma general, cuando aparece alguna de estas situaciones se pueden intentar las siguientes estrategias:

- ∗ regresar a un nodo previo e intentar una dirección diferente (máximos locales)
- ∗ realizar un gran salto en el espacio de búsqueda (mesetas)
- ∗ aplicar más de un operador antes de comparar de nuevo los valores de la función heurística (crestas)

2.3.5. Búsqueda por el mejor nodo: A*

Los métodos de búsqueda por el mejor nodo combinan algunas de las ventajas de los métodos en profundidad y en anchura. Básicamente, se trata de organizar una búsqueda mixta profundidad-anchura guiada hacia el nodo más prometedor por una función heurística que se aplica a cada paso, con independencia de la rama del árbol a la que pertenezca el nodo en cuestión. A esta familia de métodos pertenecen el algoritmo A^* y

las *Agendas* que veremos a continuación.

El algoritmo A^* trata de expandir el nodo más cercano a la meta, de entre los nodos que se encuentran en las rutas menos costosas que parten del estado inicial. Para ello, utiliza una función de evaluación f definida como

$$f(n) = g(n) + h(n)$$

donde $g(n)$ es el coste de la ruta que va del nodo de partida al nodo actual n, calculada como la suma de los costes de cada una de las acciones individuales que se emprenden a lo largo de la ruta y $h(n)$ es la función heurística, que proporciona una estimación del coste mínimo adicional de llegar desde el nodo actual n hasta el nodo meta. Nótese que g no es una estimación, se conoce exactamente su valor.

El algoritmo A* utiliza un grafo dirigido para representar el E.E. En su implementación, hace uso de las listas de nodos `ABIERTOS` y `CERRADOS`. Básicamente se procede por pasos, expandiendo un nodo en cada paso hasta que se genere un nodo que corresponda a un estado meta. En cada paso, se toman los nodos más prometedores que se han generado hasta el momento, pero que no se han expandido (`ABIERTOS`). Se generan los sucesores del nodo elegido, se les aplica la función de evaluación y se elige de nuevo el siguiente nodo a expandir. Cada nodo n en el grafo mantendrá enlaces a todos sus sucesores, pero sólo a un único predecesor: aquél que se encuentre en el camino óptimo para llegar desde el nodo inicial hasta el estado n.

El procedimiento `Propagar_Mejora(VIEJO)` se utiliza para propagar la mejora obtenida sobre el coste del nodo apuntado por VIEJO. VIEJO apunta a sus sucesores. Cada sucesor, a su vez, apunta a sus sucesores, y así sucesivamente. Por tanto, para propagar el nuevo coste hacia abajo, podemos hacer una búsqueda en profundidad, empezando en VIEJO, cambiando el valor g de cada nodo (y por tanto su valor f). La propagación termina cuando se alcanza o bien un nodo sin sucesores, o bien un nodo para el que ya se ha encontrado un camino equivalente o mejor[15]. Es fácil examinar esta condición: el enlace paterno de cada nodo apunta hacia atrás a su mejor predecesor conocido, de modo que conforme propagamos a un nodo siguiente debemos mirar si su predecesor apunta al nodo del que estamos viniendo y, si lo hace, continuar, pero si no lo hace, entonces su valor g ya refleja el mejor camino del que forma parte así que la propagación debe parar ahí. Claro que también es posible que al propagar un nuevo valor de g hacia abajo el camino que estamos siguiendo se vuelva mejor para un nodo que el camino a través del antecesor actual. Por eso debemos comparar los dos y si el camino a través del antecesor actual es aún mejor, detener también la propagación, pero si el camino a través del cual estamos propagando es mejor que el camino del antecesor actual debemos cambiar el antecesor y continuar la propagación.

Es conveniente realizar algunas observaciones sobre el papel de las funciones g, h y f en

[15]Esta segunda comprobación garantiza que el algoritmo acabará aunque haya ciclos en el grafo. Si hay un ciclo, la segunda vez que visitemos un nodo veremos que el camino no es mejor que la primera vez que lo visitamos.

```
1. Poner el nodo inicial en ABIERTOS;
   asignarle g(n)=0 y el h(n) que corresponda;
   calcular f(n);
   inicializar CERRADOS a lista vacía
2. Si ABIERTOS está vacía,
     informar del fallo y terminar
3. Elegir de ABIERTOS el nodo con mejor valor de f;
   llamarle MEJORNODO;
   quitarlo de ABIERTOS y meterlo en CERRADOS
4. Si MEJORNODO es meta,
     salir e informar de la solución siguiendo
     los enlaces que llevan del nodo meta al inicial
5. Expandir MEJORNODO generando todos sus sucesores
   Si no tiene sucesores,
     ir a 2
6. Para cada sucesor
   6.1. Poner SUCESOR apuntando a MEJORNODO
   6.2. Calcular g(SUCESOR)=g(MEJORNODO)+coste(MEJORNODO,SUCESOR)
   6.3. Si SUCESOR está en ABIERTOS,
          llamar a este nodo VIEJO;
          añadir VIEJO a la lista de sucesores de MEJORNODO;
          Si g(VIEJO)>g(SUCESOR),
            hacer que el enlace paterno de VIEJO
              apunte a MEJORNODO;
            hacer g(VIEJO)=g(SUCESOR);
            calcular f(VIEJO)=g(SUCESOR)+h(VIEJO);
            eliminar SUCESOR
   6.4. Si SUCESOR está en CERRADOS,
          llamar a este nodo VIEJO,
          añadir VIEJO a la lista de sucesores de MEJORNODO;
          Si g(VIEJO)>g(SUCESOR),
            hacer que el enlace paterno de VIEJO
              apunte a MEJORNODO;
            hacer g(VIEJO)=g(SUCESOR);
            calcular f(VIEJO)=g(SUCESOR)+h(VIEJO);
            eliminar SUCESOR;
            propagar el nuevo mejor coste realizando
              un recorrido en profundidad a partir de VIEJO
              /* Propagar_Mejora(VIEJO) */
   6.5. Si SUCESOR no está ni en ABIERTOS ni en CERRADOS,
          calcular h(SUCESOR) y f(SUCESOR),
          introducirlo en ABIERTOS
          añadirlo a la lista de sucesores de MEJORNODO

-----------------------------------------------------------------------
coste(MEJORNODO, SUCESOR)=coste de aplicar el operador que nos lleva
                          de MEJORNODO a SUCESOR
cuando SUCESOR existe en ABIERTOS, es que el nodo ya existía y lo que
       se hace es actualizarlo como hijo de MEJORNODO
si g(VIEJO)>g(SUCESOR) debemos decidir si el nuevo padre que hemos
            encontrado para el nodo es mejor que el que tenía, y si
            es así actualizar el coste y el enlace paterno, de modo
            que en el grafo queden sólo caminos óptimos
  propagar el nuevo mejor coste es necesario cuando acabamos de
           encontrar un mejor camino a VIEJO (se propaga la mejora a
           todos sus sucesores)
```

Cuadro 2.7: Algoritmo de Búsqueda A^*.

```
1. Para cada SUCESOR Nj de VIEJO
   1.1. Si el puntero al padre apunta a VIEJO,
          actualizar g(Nj)=g(VIEJO)+coste(VIEJO,Nj);
          actualizar f(Nj)
        1.1.1. Si Nj está en CERRADOS,
                 Propagar_Mejora(Nj)
   1.2. Si el puntero al padre no apunta a VIEJO,
        1.2.1. Si g(VIEJO)<g(padre de Nj),
                 poner como padre de Nj a VIEJO;
                 g(Nj)=g(VIEJO)+coste(VIEJO,Nj)
                 actualizar f(Nj)
               1.2.1.1. Si Nj está en CERRADOS,
                          Propagar_Mejora(N_j)
```

Cuadro 2.8: Algoritmo `Propagar_Mejora(VIEJO)`.

el algoritmo A*. La función g nos permite escoger el nodo a expandir sobre la base, no sólo de cuán bueno es el nodo en sí mismo (medido por h), sino también sobre la base de cuán bueno era el camino hasta el nodo. Al incorporar g en f, por tanto, no siempre elegiremos como nuestro siguiente nodo a expadir el nodo que parece más cercano a la meta. Esto es útil si nos interesa minimizar el coste del camino solución, por ejemplo. Pero si sólo nos importa llegar a una solución de la forma que sea, podemos definir siempre g como 0. Si queremos encontrar un camino que tenga el menor número de pasos, entonces debemos asignar una constante, usualmente 1, al coste de ir desde un nodo a su sucesor. Si, por otra parte, queremos encontrar el camino de menor coste y unos operadores cuestan más que otros, el coste de ir de un nodo a otro tendrá que reflejar los costes de los operadores. Así pues, el algoritmo A* puede usarse tanto si estamos interesados en encontrar un camino de coste total mínimo, como si simplemente queremos encontrar cualquier camino de la forma más rápida posible.

En cuanto a h, si es un estimador perfecto del coste del camino hasta la meta, entonces A* convergerá inmediatamente hacia la meta sin búsqueda. Si h siempre es 0, la búsqueda estará controlada por g. si g también es 0, la estrategia de búsqueda se realizará al azar. Si g siempre es 1, se realizará una búsqueda en anchura.

El algoritmo A* es *completo* y *admisible*. Es completo porque siempre acaba encontrando un camino solución, si existe. Es admisible porque para cualquier grafo, termina siempre obteniendo un camino óptimo desde un estado inicial hasta un estado meta, con tal de que exista alguno de estos caminos. Se puede demostrar que el algoritmo A* es admisible siempre que la función heurística lo sea, es decir, que raramente encontrará una solución cuyo coste sea δ mayor que el de la solución óptima si la estimación de la función heurística sobreestima el verdadero coste del camino sólo en δ.

Por último, es importante resaltar que el algoritmo A* no tiene por qué resultar adecuado para cualquier problema de búsqueda y su utilidad se deberá evaluar dentro del contexto de cada problema. Por ejemplo, en un caso en el que interesase el camino que que-

da por recorrer, sería más adecuado un algoritmo como el de ascensión a colinas que, más que realizar la búsqueda global de forma efectiva, escoge bien entre los estados sucesores del nodo actual.

2.3.6. Búsqueda por el mejor nodo: Agendas

Hasta ahora se ha asumido que el hecho de que varios caminos lleven de modo independiente al mismo estado no refuerza el mérito de ese estado. Sin embargo, esto no siempre se cumple: existen situaciones en las que no existe una única y simple función heurística que mida la distancia entre un nodo dado y un objetivo. Además, el que distintos caminos recomienden el cambio a un mismo estado mediante la realización de una tarea, puede ser importante si cada uno de ellos proporciona una razón de por qué ese estado puede conducir a resolver el problema. Cuantas más razones haya, aumenta la posibilidad de que la tarea lleve a la solución. En estos casos, es necesaria alguna estructura que nos permita almacenar las tareas propuestas, junto a las razones que se han propuesto para ellas, y algún mecanismo que nos permita gestionar este conjunto.

Las *agendas* son, básicamente, listas de tareas que puede (o debe) realizar un sistema. Cada una de las tareas de la agenda suele llevar asociada una lista de razones por las cuales se presume que acometer esa tarea es conveniente (que serán utilizadas para elaborar justificaciones y explicar procesos), y un valor que representa el peso total de la evidencia que sugiere que la tarea es útil (así, una agenda puede considerarse como un conjunto de tareas organizadas por pesos[16] –prioridades–).

El **método de búsqueda conducido por agendas** es un procedimiento por el mejor nodo en el que debe elegirse la mejor tarea de la agneda y ejecutarse, asignando para ello los recusos necesarios. El término *tarea* puede tener distintos significados: puede entenderse como una descripción explícita de lo que debe hacerse a continuación o simplemente ser una mera indicación acerca de cuál debe ser el siguiente nodo a expandir. Por otra parte, una misma tarea puede llevar asociadas distintas justificaciones, y no todas ellas han de "pesar" lo mismo.

Este método, sin embargo, provoca el gasto de una gran cantidad de tiempo en mantener ordenada la agenda, de modo que en la práctica se utiliza una estrategia modificada que compara los nuevos valores de las tareas sólo con algunos elementos superiores de la agenda (entre cinco y diez, normalmente) y si es mejor se inserta el nodo en su lugar adecuado en cabeza de la lista y si no se deja donde estaba o simplemente se inserta al final de la agenda (de vez en cuando se recorre la agenda y se reordena). Esta estrategia puede hacer que se ejecute ocasionalmente una tarea que no sea la mejor, pero su coste es significativamente menor que el del algoritmo original.

Existen algunos dominios de problemas en los que no es apropiado utilizar un mecanismo de agenda, por ejemplo, en sistemas de razonamiento no monótono en los que las razones y justificaciones que apoyan en un momento dado una tarea pueden no mantenerse un tiempo después.

[16]Este orden permitiría, además, la inserción ordenada de nuevas tareas; si las justificaciones de una tarea cambian, el peso debe ser recalculado y puede que tenga que ser trasladada a una nueva localización en la lista.

```
Hasta que se alcance un estado objetivo o la agenda esté vacía hacer:
1. Identificar la mejor tarea de la agenda (MEJORNODO)
2. Ejecutarla
3. Si se han generado nuvas tareas,
     para cada una de ellas
       3.1. Si no estaba ya en la agenda,
              añadirla a la lista
       3.2. Si ya estaba en la agenda,
              si no tiene la misma justificación,
                añadir la nueva justificación
4. Si se ha añadido una tarea o una justificación,
     calcular el peso asignado a estas tareas combinando
     la evidencia de todas sus justificaciones y
     recomponer la agenda
```

Cuadro 2.9: Algoritmo de Búsqueda conducida mediante *Agenda*.

Capítulo 3

Representaciones Formales del Conocimiento

Resulta evidente que si un programa de IA debe explotar eficazmente un conjunto determinado de conocimientos de un dominio concreto, al menos parte de la potencia de dicho programa vendrá determinada por la eficacia y la consistencia del esquema que hayamos elegido para *representar el conocimiento*.

3.1. Aspectos Generales de la Representación del Conocimiento

En cualquier dominio de aplicación nos encontraremos siempre dos tipos de entidades diferentes:

- *hechos* o *verdades del dominio* y
- *representaciones* de los hechos[1]

Para manipular computacionalmente "hechos" necesitamos definir un conjunto de procedimientos que los conviertan en representaciones. Una vez ejecutado nuestro programa de IA, necesitamos nuevos procedimientos que conviertan las representaciones internas en hechos comprensibles para nosotros. Este proceso configura lo que se denomina *fase de codificación-decodificación*[2].

Cualquier procedimiento para representar conocimiento tiene que reunir un conjunto mínimo de condiciones:

Transparencia Concepto que hace referencia a si podemos o no identificar fácilmente el conocimiento representado.

[1] Las estructuras internas que manipulan los programas de IA y que se corresponden con las verdades del dominio.

[2] En ambas se va a producir una inevitable pérdida de información.

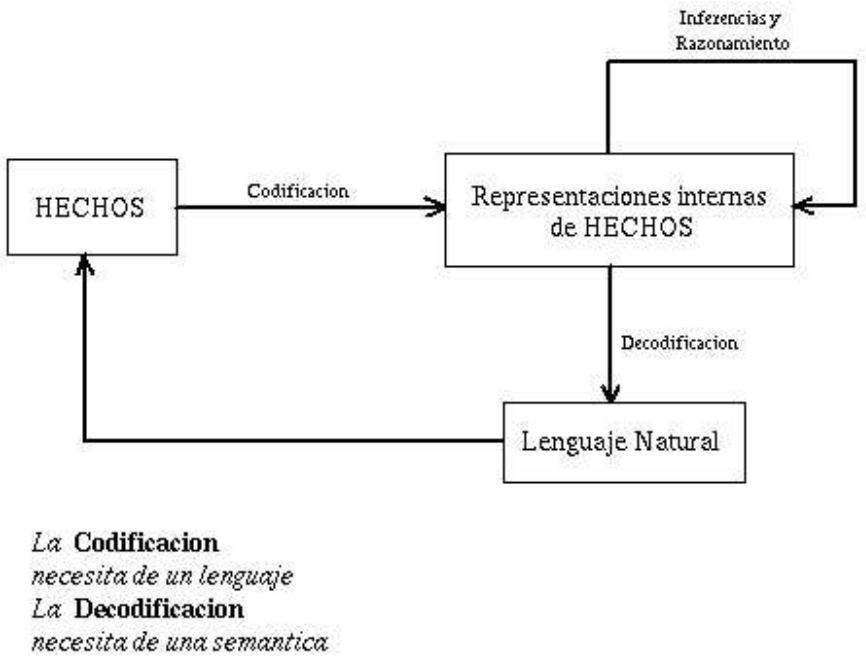


Figura 3.1: Ciclo básico de codificación-decodificación.

Naturalidad Significa si podemos o no representar el conocimiento en su forma original.

Claridad Referencia si podemos o no representar directamente el conocimiento.

Eficiencia O facilidad relativa con la que se puede acceder a conocimientos específicos durante la ejecución.

Adecuación O capacidad del esquema de representación para representar todos los conocimientos y tipos de conocimiento que requiere el sistema.

Modularidad O capacidad del esquema de representación para fragmentar los distintos tipos de conocimiento[3] del sistema.

Un aspecto importante a tener en cuenta es la flexibilidad con la que podemos manejar el conocimiento. Como regla general, cuanto mayor sea el nivel del conocimiento barajado, menor será su flexibilidad: el conocimiento de alto nivel es potente pero muy poco flexible, mientras que el conocimiento de bajo nivel des flexible pero poco potente.

Los esquemas de representación del conocimiento existentes pueden clasificarse en las siguientes categorías:

[3]El conocimiento involucrado en los distintos procesos de razonamiento puede estar detallado de muy diversas formas: hablamos de "principios primarios" o bloques de construcción básicos sobre los que se basa el dominio en cuestión y a partir de los que deben poder ser desarrollados otros principios más específicos, nuevos teoremas y reglas de acción, base a su vez para derivar conocimientos adicionales.

- √ *Métodos Declarativos*, hacen énfasis en la representación del conocimiento como una acumulación o colección de hechos estáticos a los que se añade cierta información limitada que describe cómo se va a emplear el mencionado conocimiento, esto es, para cuya manipulación se define un conjunto genérico y restringido de procedimientos. Presentan la ventaja de que las verdades del dominio se almacenan una sola vez y es fácil incrementar e incorporar nuevo conocimiento sin modificar ni alterar el ya existente.
- √ *Métodos Procedimentales*, que enfatizan la representación del conocimiento en forma de estructuras dinámicas (procedimientos, en los que la representación de los hechos va implícita) que describen procedimienos de utilización de los conocimientos. Sus ventajas son, por ejemplo, que al dar prioridad a los procedimientos sehace mayor énfasis en las capacidades inferenciales del sistema, que permiten explorar distintos modelos y técnicas de razonamiento, que permiten trabajar con falta de información y con datos de carácter probabilístico y que incorporan de forma natural conocimiento de tipo heurístico.

En realidad, los problemas interesantes de IA suelen requerir distintas proporciones de ambas filosofías en la representación del conocimiento del dominio.

3.2. Lógica de Proposiciones y Lógica de Predicados

La **lógica de proposiciones** y la **lógica de predicados** suelen englobarse en lo que denominamos **lógica formal**. Como esquema de representación del conocimiento, la *lógica formal* permite derivar conocimiento[4] a partir de conocimiento ya existente a través de procesos deductivos, esto es, permite afirmar que una aseveración es cierta si puede deducirse a partir de otras que se sabe que son ciertas.

La más sencilla de las lógicas formales es la *lógica de proposiciones*, en la que los hechos del mundo real se representan como proposiciones lógicas (*fórmulas bien definidas*, FBD, o *fórmulas bien formadas*, FBF), por ejemplo: "Sócrates es un hombre" se denotaría SOCRATESHOMBRE y "Aristóteles es un hombre" sería ARISTOTELESHOMBRE. Está claro que la *lógica de proposiciones* presenta varios problemas, como la representación eficaz de varios ejemplos de una misma entidad y la cuantificación, no es una representación versátil.

Surge la necesidad, pues, desde la óptica formal, de utilizar la *lógica de predicados*, que representa el conocimiento como declaraciones lógicas[5] que son FBF o FBD. Así, el ejemplo anterior se convierte en HOMBRE(SOCRATES).

Los componentes básicos de un esquema de representación del conocimiento basado en lógica de predicados son:

[4] Aunque no es *nuevo* conocimiento, sino sólo conocimiento que no estaba representado, explicitado.

[5] Distinción *proposición-declaración*: una declaración es una proposición que permite cuantificación o ejemplificación.

- ▷ Alfabeto
- ▷ Lenguaje formal
- ▷ Conjunto de enunciados básicos o axiomas
- ▷ Reglas inferenciales

Los axiomas describen fragmentos de conocimiento y las reglas inferenciales se aplican a los axiomas para tratar de deducir nuevos enunciados verdaderos.

3.2.1. Alfabeto

En cualquier *lenguaje formal*, el **alfabeto** es el conjunto de símbolos a partir de los que se construyen los enunciados.

En lógica de predicados el alfabeto está constituido por:

- ◦ *Predicados*, representan relaciones en el dominio de discurso y pueden tomar dos valores: "verdadero" o "falso". Un predicado es verdadero si los elementos involucrados verifican la relación especificada. Los predicados y los términos que identifican los elementos relacionados se utilizan para definir las *fórmulas atómicas* o *átomos* (unidades estructurales mínimas que se pueden definir), por ejemplo: HOMBRE(JUAN), MASALTO(JUAN,PEPE) o MASALTO(JUAN,PADRE(PEPE)).
- ◦ *Variables*, conjuntos de constantes.
- ◦ *Funciones*, describen elementos y los identifican como resultado único de la aplicación de una transformación entre otros elementos del dominio, por ejemplo: PADRE(JUAN), MADRE(PADRE(JUAN)) o ASESINO(X).
- ◦ *Constantes*, representan elementos específicos del dominio de discurso.
- ◦ *Juntores*, elementos gramaticales del lenguaje que permiten representar declaraciones compuestas:
  - ∗ *AND*, hace que para que una FBD sea cierta todos y cada uno de los componentes relacionados tengan que serlo.
  - ∗ *OR*, hace que si al menos uno de los componentes relacionados es cierto, la FBD correspondiente lo sea.
  - ∗ *NOT*[6], cambia el estado lógico deuna expresión.
  - ∗ →, establece relaciones de implicación entre expresiones, siendo la construcción A → B equivalente a NOT A OR B.
  - ∗ =, indica la equivalencia lógica entre dos FBD.
- ◦ *Cuantificadores*, elementos que resuelven uno de los problemas mencionados en la lógica de proposiciones:

[6]En realidad, el NOT es un operador lógico, no un juntor. La diferencia entre ambos estriba en que un operador lógico sólo cambia la entrada, no establece una relación, funciona como un modificador, mientras que un juntor relaciona dos elementos a través de un operador lógico.

  - *Universal* $\forall x$, establece que la FBD es cierta para todos los valores que puede tomar x.
  - *Existencial* $\exists x$, establece que existe al menos un x que hace verdadera la FBD.

  En ambos casos la variable genérica x asociada al cuantificador se denomina *variable cuantificada*, mientras que el *alcance* del cuantificador es la FBD que le sigue.

- *Delimitadores*, elementos como "," y "()", necesarios para obtener representaciones correctas del conocimiento.

Con predicados, variables, funciones y constantes pueden construirse fórmulas atómicas susceptibles de una adecuada representación. Juntores, cuantificadores y delimitadores aportan la *semántica* necesaria para dotar de significado al conjunto específico de símbolos definidos en el dominio y establecer su correspondencia en el universo de los hechos.

3.2.2. Lenguaje formal

El **lenguaje formal** asociado a la lógica de predicados es el conjunto de tods las FBD que se pueden construir legalmente[7] a partir del alfabeto.

Se puede definir inductivamente una FBD del siguiente modo:

- Cualquier fórmula atómica es FBD.
- Si F y G son FBD, entonces también lo son NOT F, F AND G, F OR G y F→G.
- Si x es una variable y F es una FBD, entonces también son FBD $(\forall x)F$ y $(\exists x)F$.

El conjunto de FBD que seamos capaces de construir sobre un dominio concreto constituye el *lenguaje formal* asociado.

Existen, sin embargo, expresiones sencillas que, de acuerdo con la definición anterior, no pueden ser consideradas FBD, como por ejemplo NOT F(A), $(\forall P)P(A)$ o $(\exists f)f(A)$. Estas expresiones no son FBD en lógica de predicados. Las lógicas que no permiten cuantificación sobre predicados o funciones se denominan *lógicas de primer orden.*

3.2.3. Reglas de inferencia

La *inferencia* en lógica formal es el proceso de ejemplificación que permite generar nuevas FBD a partir de FBD ya existentes, mediante la aplicación de las llamadas **reglas de inferencia**. De tales reglas, la más común es la llamada *modus ponens* que se puede expresar:

$$[P1 \text{ and } (P1 \rightarrow P2)] \rightarrow P2$$

[7] Reflejando fielmente el dominio que se quiere representar, de acuerdo con su estructura.

que no es sino la formalización del razonamiento deductivo. Otra regla de inferencia común es la *especialización universal*:

$$\text{INDIVIDUO and } (\forall x)[f(x)] \rightarrow f(\text{INDIVIDUO})$$

Además de estas dos reglas básicas de inferencia existen otras, como la *sustitución*, el *modus tollens*, etc., todas ellas capaces de generar nuevas FBD a partir de FBD ya existentes. Con estas reglas, con tal de que los axiomas construidos sean válidos, la inferencia sintáctica es siempre posible. Desgraciadamente, que una inferencia sea posible no quiere decir que el resultado tenga o no el menor interés. Por el contrario, en lógica formal es muy común que las inferencias realizadas conduzcan a información absolutamente irrelevante y, además, no se nos asegura la obtención de nueva información útil en un tiempo prudencial.

3.3. Ingeniería del Conocimiento y Lógica Formal

Una de las tareas de la **Ingeniería del Conocimiento** es estructurar y codificar conocimiento para que éste sea utilizado de forma eficiente por un programa de IA. Veremos a continuación algunas de sus características fundamentales en relación con la lógica formal.

Cuando empleamos un esquema de representación del conocimiento basado en lógica formal, el proceso básico de ingeniería del conocimiento consta de las siguientes fases que constituyen una metodología de diseño y desarrollo:

1. *Identificación*: Comprensión e identificación del conocimiento relevante del dominio.
2. *Formalización*: Formalización de los enunciados correspondientes.
3. *Descomposición*: Análisis o fragmentación de los enunciados en sus partes constituyentes.
4. *Traducción*: Establecimiento de la simbología adecuada para representar elementos y relaciones.
5. *Recomposición*: Construcción de las FBD.

Estos cinco pasos configuran la base de *codificación* mediante la cual pretendemos obtener representaciones del conocimiento del dominio manejables desde una perspectiva computacional[8]. Podemos ver un ejemplo en la tabla 3.1 (página 45).

En este proceso, aparecen frecuentemente problemas de interpretación, ante los que debemos adoptar una postura desde la cual nos interesa siempre encontrar estructuras lo más simétricas posibles. Las transformaciones usualmente provocarán la pérdida de matices: en ocasiones será una pérdida aceptable, pero las expresiones correspondientes no

[8]Puesto que, como sabemos, necesitamos que la información esté preprocesada para usarla en IA; no todo el conocimiento de un dominio será relevante, ni será siempre el mismo.

```
Declaración: "Milú es un perro foxterrier blanco"

(1) Descomposición: Milú es un perro
                    Milú es un foxterrier
                    Milú es blanco
(2) Traducción: perro(Milú)
                foxterrier(Milú)
                blanco(Milú)
(3) Recomposición: perro(Milú) and foxterrier(Milú) and blanco(Milú)
```

Cuadro 3.1: Ejemplo de codificación de una expresión.

podrán ser consideradas equivalentes. Se plantearán cuestiones como: ¿podemos prescindir de elementos de la frase original y sin embargo obtener una FBD apropiada? ¿qué es lo verdaderamente informativo? ¿podemos eliminar tiempos verbales, sujetos en tercera persona, primera? ¿podemos asumir relaciones causa-efecto no evidentes como implicaciones? ¿podemos perder matices cuantitativos? En la mayoría de los casos, dependerá del contexto: la misma información en distintos contextos puede ser considerada de diferente manera.

Al margen de problemas de interpretación intrínsecamente asociados a la semántica, existen otros problemas derivados de la interpretación de frases en su contexto. Por ejemplo, consideremos las dos siguientes frases: "A todo el mundo le gusta el arte o el deporte" y "En los EE.UU. todos son o republicanos o demócratas". Ambas son estructuralmente idénticas pero presentan importantes diferencias conceptuales debidas a cuestiones de contexto. Su traducción debe efectuarse teniendo en cuenta (y especificando) la naturaleza del juntor OR:

```
OR exclusivo:
              (Vx)[persona(x) -> gusta_arte(x) or gusta_deporte(x) or
                                 [gusta_arte(x) and gusta_deporte(x)]]
              (Vx)[estadounidense(x) -> republicano(x) or democrata(x)

OR inclusivo:
              (Vx)[persona(x) -> gusta_arte(x) or gusta_deporte(x)]
              (Vx)[estadounidense(x) -> republicano(x) or democrata(x)
                                        and not [republicano(x) and
                                        democrata(x)]]
```

Dependiendo de las fases de identificación y formalización, a la hora de codificar conocimiento en lógica formal deberemos decidir qué carácter le queremos dar al juntor OR para obtener las mejores representaciones en el dominio. Lo que está claro es que, en una

misma aplicación, no se pueden mezclar juntores OR de distinta naturaleza[9].

Empero, siempre deberemos ser capaces de encontrar una traducción en lenguaje formal satisfactoria para un determinado enunciado en lenguaje natural.

En la fase de decodificación se hace uso de la semántica para interpretar los enunciados de las correspondientes FBD. Igual que la codificación, la decodificación suele ir siempre acompañada de cierta imprecisión, ambigüedad o inexactitud. Los esfuerzos deben dirigirse hacia la minimización de la imprecisión asociada a los lenguajes formales.

3.4. Evaluación y Resolución en Lógica Formal

Hemos visto que la lógica formal permite derivar declaraciones nuevas, y ciertas, a partir de un conjunto de axiomas o principios básicos (las verdades en nuestro dominio de discurso). En cualquier caso, la verdad de una declaración está relacionada con el proceso de interpretación. En lógica de proposiciones la verdad de una proposición compleja puede determinarse a partir de los valores de la tabla 3.2 (página 46).

X	Y	X AND Y	X OR Y	X ->Y	NOT X	X = Y
sí	sí	sí	sí	sí	no	sí
sí	no	no	sí	no	no	no
no	no	no	no	sí	sí	sí
no	sí	no	sí	sí	sí	no

Cuadro 3.2: Tabla de Verdad.

En lógica formal, la determinación de la verdad de una fórmula compleja supone la reducción sucesiva de la declaración, desde dentro hacia afuera, utilizando convenientemente la tabla de verdad. Este procedimiento de evaluación es muy sensible a la dificultad de los dominios, y se complica enormemente aún en dominios relativamente sencillos, aunque con frecuencia ello es debido a deficiencias en la fase de codificación, donde la ambigüedad del lenguaje natural, la multitud de representaciones más o menos equivalentes y la carencia de sentido común conducen muchas veces a callejones sin salida.

El método propuesto para evaluar expresiones no es muy operativo computacionalmente, ya que nos obliga a realizar diversas sustituciones. Lo ideal sería disponer de un procedimiento de demostración que llevase a cabo, en una única operación, la variedad de procesos involucrados en un razonamiento basado en declaraciones de la lógica formal. Surgen así los **procedimientos de resolución**, que operan sobre declaraciones previamente normalizadas.

La *resolución* obtiene demostraciones por medio de la denominada *refutación*, prueba consistente en tratar de encontrar que la negación de una declaración produce una contradicción axiomática. Este enfoque es radicalmente diferente a la técnica hasta ahora empleada de *demostración hacia atrás* (desde los teoremas hasta los axiomas).

[9] Al menos denominándolos con el mismo símbolo.

De todas formas, antes de aplicar la *resolución por refutación* hemos de tener todo nuestro conocimiento estructuralmente *normalizado*. Para ello, trataremos de construir la denominada *forma normalizada conjuntiva de Davis* que, en esencia, trata de simplificar las FBD y separar los cuantificadores del resto de la fórmula. El procedimiento para obtenerla es:

1. Eliminar las implicaciones.
2. Reducir el número de negaciones.
3. Normalizar las variables para que cada cuantificador esté ligado a una única variable.
4. Obtener la fórmula normalizada *prenex*, constituida por un prefijo de cuantificadores seguido por una matriz libre de cuantificadores.
5. Eliminar los cuantificadores existenciales.
6. Abandonar el prefijo.
7. Convertir la matriz en una conjunción de disyunciones.
8. Identificar las cláusulas.
9. Normalizar las variables por separado en el conjunto de cláusulas generadas en el paso anterior.

Para realizar las transformaciones necesarias para obtener la forma normalizada conjuntiva de Davis es útil considerar las equivalencias de la tabla 3.3 (página 47).

$P1 \rightarrow P2$	not $P1$ or $P2$
$P1$ or ($P2$ and $P3$)	($P1$ or $P2$) and ($P1$ or $P3$)
$P1$ and ($P2$ or $P3$)	($P1$ and $P2$) or ($P1$ and $P3$)
$P1 \rightarrow P2$	not $P2 \rightarrow$ not $P1$
not (not $P1$)	$P1$
not ($P1$ or $P2$)	not $P1$ and not $P2$
not ($P1$ and $P2$)	not $P1$ or not $P2$
not $\forall x P(x)$	$\exists x$ not $P(x)$
not $\exists x P(x)$	$\forall x$ not $P(x)$
$P1$ or $Falso$	$P1$
$P1$ or $Verdadero$	$Verdadero$
$P1$ and $Falso$	$Falso$
$P1$ and $Verdadero$	$P1$
$P1$ or not $P1$	$Verdadero$
$P1$ and not $P1$	$Falso$

Cuadro 3.3: Tabla de Equivalencias.

La eliminación de cuantificadores existenciales es posible ya que debe existir al menos un valor que pueda sustituir a la variable cuantificada existencialmente, y que haga verdadera la fórmula. Así, podemos eliminar el cuantificador sustituyendo la variable por

una referencia a una función que genere el valor deseado. Dado que no siempre conocemos el valor que hace verdadera la fórmula, debemos crear un nuevo nombre de función para cada sustitución. De este modo no hacemos ninguna afirmación sobre tales funciones y sólo indicamos que deben existir. Por ejemplo:

$$(\exists y)\text{PRESIDENTE}(y) = \text{PRESIDENTE}(S1)$$
$$(\forall x)(\exists y)\text{PADRE}(y, x) = (\forall x)\text{PADRE}(S2(x), x)$$

Tales funciones se denominan *funciones de Skolem* si tienen argumentos y *constantes de Skolem* si no los tienen.

Una vez conseguida una fórmula *prenex* en la que los únicos cuantificadores sean universales, puede prescindirse del prefijo sin que por ello la declaración se vea afectada. Para convertir la matriz resultante en una conjunción de disyunciones tendremos que emplear las propiedades *asociativa* y *distributiva*:

$$P1 \text{ or } (P2 \text{ or } P3) = (P1 \text{ or } P2) \text{ or } P3$$
$$(P1 \text{ and } P2) \text{ or } P3 = (P1 \text{ or } P3) \text{ and } (P2 \text{ or } P3)$$

Una vez obtenidas e identificadas las cláusulas correspondientes, la normalización de variables por separado supone renombrar las variables de forma que no haya dos cláusulas que hagan referencia a la misma variable. El resultado final es una disyunción de *literales*.

Obtenida la forma normalizada conjuntiva de Davis, el método de resolución por refutación no es más que un proceso iterativo simple en el que, en cada paso, se resuelven dos cláusulas llamadas *cláusulas padres* produciéndose una nueva cláusula inferida[10]. El proceso continúa hasta que encontremos una cláusula vacía (indicativa de una contradicción) o bien no se pueda seguir. El hecho de llegar a una contradicción partiendo de un conjunto de axiomas ciertos a los que se ha incorporado la negación de la hipótesis que queremos verificar, es indicativo de que la hipótesis debe ser cierta.

3.5. Introducción a otras Lógicas

Las lógicas formales de primer orden son útiles para encarar problemas en una amplia variedad de dominios. Sin embargo, hay multitud de campos interesantes en los que, simplemente, los esquemas de representación del conocimiento basados en lógica formal de primer orden son totalmente inadecuados, como por ejemplo en casos en los que la información contiene grados relativos de magnitudes, es incierta o imprecisa, posee heurísticas amplias, etc.

Para tratar de resolver este tipo de problemas han sido propuestos diversos enfoques, entre los que cabe considerar:

- ⋆ Lógicas no monótonas
- ⋆ Razonamiento probabilístico

[10]Dicho coloquialmente, buscaremos pares de proposiones $(P, \neg P)$ en la disyunción de literales final con el propósito de irlos descartando.

- ⋆ Lógica difusa
- ⋆ Modelos de credibilidad o modelos cuasiestadísticos
- ⋆ Razonamiento por concepto
- ⋆ Tratamiento de incertidumbre

Por ejemplo, la lógica no monótona permite la eliminación del conocimiento. De este modo, la verosimilitud de una afirmación puede estar basada en la falta de confianza en alguna otra afirmación.

Los sistemas tradicionales basados en lógica formal son monótonos en el sentido de que el número de declaraciones verdaderas se incrementa estrictamente en el transcurso de los procesos inferenciales (otra cosa es que esa nueva información resulte útil). Pueden añadirse nuevas declaraciones al sistema y demostrarse nuevas relaciones, y ello nunca invalidará una declaración demostrada o conocida previamente. Esto hace que cuando se añade una nueva declaración no sea necesario realizar ningún análisis de consistencia y que, dada una declaración que acaba de ser demostrada, no sea necesario recordar las declaraciones en las que se ha basado dicha demostración, ya que no hay riesgo de que desaparezcan.

Desgraciadamente, lo sistemas monótonos tienen problemas cuando trabajan con información incompleta, entornos cambiantes y generación de supuestos, situaciones todas ellas típicas de problemas del mundo real.

Los sistemas no monótonos basan su estrategia en el hecho de que, normalmente, nunca se dispone de toda la información que es útil para resolver un problema. Sin embargo, cuando dicha información falta, siempre pueden hacerse suposiciones sensatas mientras no se presente ninguna evidencia contradictoria.

La construcción de tales suposiciones origina lo que se denomina *razonamiento por defecto*, que es un caso particular de lógica no monótona. Su no-monotonicidad proviene del hecho de que la conclusión de una evidencia nueva puede forzar la eliminación de conocimiento hasta entonces considerado “cierto” o “válido”.

Capítulo 4

Métodos Estructurados de Representación del Conocimiento

La lógica formal permite la utilización de procedimientos de resolución que posibilitan el razonamiento con hechos. Sin embargo, los objetos del universo real tienen propiedades y se relacionan con otros objetos. Así, sería útil disponer de estructuras de representación que permitiesen, por una parte, agrupar propiedades, y por otra, obtener descripciones únicas de objetos complejos. Por otra parte, los objetos no son las únicas entidades estructuradas del universo, también sería muy útil poder representar eficazmente escenarios y secuencias típicas de acontecimientos.

Para tratar de dar respuesta a tales cuestiones, en IA se utilizan **esquemas no formales de representación del conocimiento**[1]. Estos esquemas son fundamentalmente métodos estructurados de representación, y tienen que verificar las siguientes propiedades:

Adecuación representacional El esquema elegido debe ser capaz de representar las distintas clases de conocimiento del dominio.

Adecuación inferencial El esquema elegido debe permitir la manipulación del conocimiento para obtener conocimiento nuevo.

Eficiencia inferencial El esquema debe ser versátil utilizando aquella información que permita optimizar el proceso inferencial.

Eficiencia adquisicional El esquema debe suministrar vías que permitan la incorporación de información y conocimiento nuevos, es decir, debe ser fácilmente actualizable según aumenta el conocimiento del dominio.

Independientemente del esquema de representación elegido, es muy útil considerar una serie de elementos que nos permiten establecer relaciones entre distintas estructuras de conocimiento. Tales elementos son:

ES_UN (IS_A): que permite establecer relaciones entre taxonomías jerárquicas.

[1] Que también pueden clasificarse como esquemas *declarativos* o *procedimentales* –ver sección 3.1, página 41–.

ES_PARTE_DE (PART_OF): que permite establecer relaciones entre objetos y componentes de un objeto.

Una propiedad importante de ambas relaciones es la de *transitividad*, que está vinculada al razonamiento deductivo. Además, la transitividad de la relación ES_UN nos permite establecer un método para la obtención de propiedades de los objetos relacionados, lo que configura un proceso de *herencia de propiedades* mediante el cual, si un objeto pertenece a una determinada clase, a través de la relación ES_UN dicho objeto hereda las propiedades de la clase.

Desde una perspectiva formal podemos definir una correspondencia entre ambas relaciones y la lógica de predicados, donde HOMBRE(MARCO) se convierte en ES_UN(MARCO, HOMBRE).

Los *métodos declarativos* que estudiaremos en este capítulo serán:

- √ *Redes semánticas*, que permiten describir simultáneamente acontecimientos y objetos.
- √ *Modelos de dependencia conceptual*, estructuras especializadas que proporcionan mecanismos para representar relaciones entre los componentes de una acción.
- √ *Frames*, estructuras genéricas que permiten representar objetos complejos desde diferentes puntos de vista.
- √ *Guiones*, estructuras especializadas, que derivan de las frames y que son útiles para representar secuencias comunes de acontecimientos.

Aunque básicamente diferentes, estas cuatro estructuras de representación comparten ciertos elementos: en todos ellos las entidades complejas pueden describirse como una colección de atributos y unos valores asociados (estructuras *slot-and-filler*).

También haremos en este capítulo una breve descripción del paradigma de *Orientación a Objetos* como esquema alternativo y potente de representación jerárquica de objetos y clases que comparten propiedades.

El método procedimental que estudiaremos será el de las *Reglas de Producción*.

4.1. Redes Semánticas

Las **redes semánticas** son estructuras declarativas de representación en las que el conocimiento se representa como un conjunto de *nodos* conectados entre sí por medio de *arcos* etiquetados que representan relaciones lingüísticas entre nodos.

En las *redes semánticas* cada propiedad incluye un enlace unidireccional. Para establecer enlaces bidireccionales hay que tratar cada relación por separado (como dos enlaces unidireccionales). Un enlace puede ser considerado como algo que aseveramos de un nodo en relación a otro. Puesto que una aseveración dada sólo puede ser *cierta* o *falsa*, un enlace es una relación binaria entre nodos.

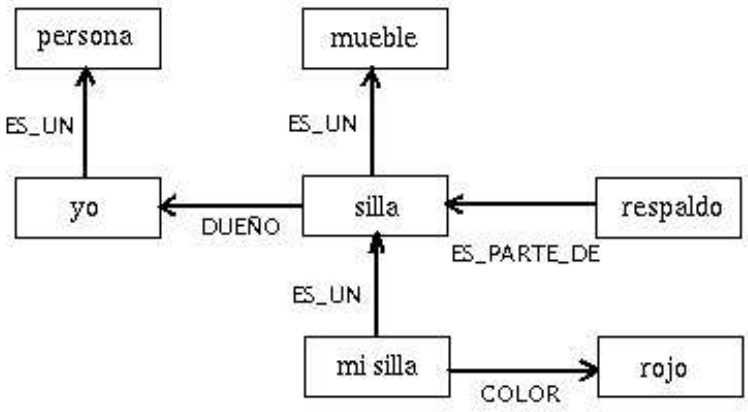


Figura 4.1: Una red semántica sencilla.

Dos de las relaciones binarias más corrientes en las redes semánticas son las relaciones ya mencionadas ES_UN y ES_PARTE_DE. En este contexto, la relación ES_UN se emplea para establecer el hecho de que un elemento dado es miembro de una clase de elementos que tienen en común un conjunto de propiedades distinguibles, aunque también puede emplearse para representar situaciones, acciones y eventos.

Las relaciones más frecuentes en redes semánticas pueden clasificarse en una de las siguientes categorías:

- Ocurrencia: cuando se relaciona un miembro de una categoría general con la categoría a la que pertenece (se suele etiquetar PERTENECE).
- Generalización: cuando se relaciona una entidad con otra de carácter más general (ES_UN –ejemplificación–).
- Agregación: cuando se relacionan componentes de un objeto con el objeto propiamente dicho (ES_PARTE_DE).
- Acción: cuando se establecen vínculos dinámicos entre diferentes objetos.
- Propiedades: que son relaciones entre objetos y características de los objetos.

Desde una perspectiva computacional, la implementación de una red semántica requiere la construcción de una tabla de n tuplas del tipo `objeto-atributo-valor` de forma que el `objeto` sea el nodo padre, el `atributo` sea el arco y el `valor` sea el nodo destino. También existe una correspondencia directa entre las redes semánticas y la lógica formal, donde los predicados tendrían un aspecto `atributo(objeto,valor)`.

En redes semánticas, la herencia de propiedades nos dice que cualquier propiedad que consideremos cierta para una clase de elementos debe ser cierta para cualquier ejemplo de la clase. Este concepto hace que las redes semánticas sean particularmente interesantes para representar dominios que se puedan estructurar como taxonomías.

En cuanto a la forma de razonar con redes semánticas, el modelo permite obtener asociaciones simplemente rastreando los enlaces del sistema, sin que ninguna regla semántica rigurosa guíe el proceso. Esto hace que, mientras que en un sistema de lógica formal, dado que las inferencias se realizan sobre la base de manejos sintácticos uniformes de símbolos, éstas sean siempre válidas (aunque en ocasiones puedan ser irrelevantes), en redes semánticas las relaciones que se representan pueden no ser totalmente rigurosas (la mayoría de las veces por *condiciones de excepción no reconocidas*) y, por lo tanto, las inferencias obtenidas por rastreo pueden no ser válidas.

Otra técnica de corriente de razonamiento con redes semánticas es el *emparejamiento*, consistente en la construcción de fragmentos de red, algunos de cuyos nodos tiene valores definidos pero otros no. En este caso los nodos sin valores se representan por variables. El sistema debe entonces tratar de encontrar un fragmento de la red semántica original que encaje perfectamente en el fragmento de red semántica que hemos construido para representar nuestro problema.

4.2. Modelos de Dependencia Conceptual

La **dependencia conceptual** intenta representar el significado de frases en lenguaje natural de forma que se posibilite la derivación de conclusiones y la representación sea independiente del lenguaje utilizado en las declaraciones originales.

Una *representación de dependencia conceptual* no se construye a partir de primitivas que correspondan a palabras concretas de una declaración dada, sino que se utilizan primitivas conceptuales que pueden combinarse para formar significados de palabras en cualquier lenguaje. La *dependencia conceptual* proporciona, pues, simultáneamente una estructura y un conjunto específico de primitivas a partir de las cuales pueden construirse representaciones de elementos concretos de información.

Este modelo facilita el razonamiento con el conocimiento representado ya que al descomponer el conocimiento en primitivas se necesitan menos reglas de inferencia, muchas inferencias están autocontenidas en la representación y la estructura inicial construida con la información disponible facilita el enfoque de atención del programa que debe entender las frases, ya que contiene huecos que deberán rellenarse[2].

4.3. Frames y Guiones

Ante un problema nuevo, nadie empieza directamente un análisis exhaustivo y desde cero para construir incrementalmente estructuras de conocimiento cada vez más complejas hasta alcanzar la que describa perfectamente la nueva situación presentada. Por el contrario, el primer paso susle consistir en recuperar experiencias anteriores y tratar de razonar "por semejanza".

[2]Pregunta de examen: *¿Cómo representaríamos en redessemánticas una dependencia conceptual?* Respuesta: Mediante un arco etiquetado.

4.3.1. Frames

Las **frames**[3] pueden describirse como redes semánticas complejas que tratan el problema de la representación desde la óptica del razonamiento por semejanza: describen clases de objetos y pueden definirse como representaciones estructuradas de conocimiento estereotipado[4].

Estructuralmente, una *frame* consta de una *cabecera*, que le da nombre y es representativa de la clase de objetos que se describen en ella, y de un conjunto de *slots*, cada uno de los cuales representa una propiedad o atributo del elemento genérico representado por la frame. Cada slot puede tener distintos slots anidados y sin limitación de profundidad, debiendo tenerse en cuenta que dicha posición del slot en la frame puede cambiar su significado, pues cada una de las identaciones de los slots representa un nivel de conocimiento o nivel epistemológico, y su contenido es una especialización del nivel anterior.

```
automóvil
    tipos
         todo_terreno
         deportivo
         utilitario
    componentes
         carrocería
         puertas
         capó
```

Cuadro 4.1: Ejemplo de *frame*.

Un sistema en el que el conocimiento esté representado por medio de frames utiliza con profusión el concepto de herencia, para lo que se emplean slots de tipo ES_UN, que permiten la entrada de información a una frame en un nivel de conocimiento determinado y partir del cual la información de la clase correspondiente pasa al objeto considerado (en ese sentido, una frame puede ser considerada de orden superior).

Las frames suelen incorporar también información procedimental. Para ello, ciertos slots llevan asociados procedimientos que la mayor parte del tiempo están inactivos, pero que cuando son activados desencadenan acciones concretas. Algunos de tales procedimientos, denominados *demons* son: IF_NEEDED, IF_ADDED, IF_REMOVED, IF_STORED, IF_RETRIEVED, etc.

Cuando un demon es activado (por una entrada en la frame, al nivel correspondiente), el procedimiento que sigue al demon es ejecutado y, normalmente, el demon, cumplida ya su misión, es eliminado inmediatamente. Los demons son estructuras muy útiles ya que proporcionan uniones procedimentales entre distintas frames, posibilitan la ejecución de rutinas externas e imprimen un cierto carácter dinámico a la representación del conocimiento con frames. Como estructuras de representación, las frames pueden emplearse

[3]La palabra inglesa *frame* puede significar *esqueleto*, *marco* o *fotograma*.

[4]Un *estereotipo* puede considerarse un paradigma (mejor modelo) estructural.

como elementos descriptivos o elementos de control del conocimiento.

Con lo visto, podemos ya resaltar que:

- Las frames permiten definir procesos de razonamiento con información incompleta.
- Las frames permiten inferir rápidamente hechos no representados de forma explícita.
- Las frames imprimen cierto carácter dinámico a la representación al definir procesos que establecen relaciones entre otras frames y conexiones con el mundo exterior.
- Las frames utilizan con profusión el concepto de herencia.

El razonamiento con frames suele comenzar con la selección de una frame determinada que se ajuste a nuestra situación actual. Dado que en la mayoría de los casos no tendremos ninguna frame que describa exactamente nuestro estado inicial, comenzaremos seleccionando la más ajustada en base a la evidencia parcial disponible. A continuación, se produce la *ejemplificación* de la frame seleccionada tras considerar ciertas condiciones específicas actuales.

En general, el proceso de ejemplificación asocia un individuo particular con una clase, es decir, obtenemos una descripción individual del problema actual considerando, por un lado, la descripción de la clase genérica y, por otro lado, las características específicas actuales.

Este procedimiento, aunque potente, está siempre sujeto y matizado por la experiencia individual, de modo que en ocasiones la subjetividad en la percepción desvirtúa la calidad de la representación.

4.3.2. Guiones

Los **guiones** son especializaciones del concepto general de *frame*, que conforman estructuras capaces de representar prototipos de secuencias de sucesos. En los guiones el tiempo es siempre una variable implícita.

Los elementos estructurales más habituales de los guiones son:

- Las condiciones de entrada o condiciones iniciales de aplicabilidad del guión.
- Los resultados o hechos que serán verdaderos una vez el guión sea ejecutado.
- Las herramientas u objetos relevantes en el desarrollo del guión.
- Los papeles o roles que representan los actores que actúan en el guión.
- Las escenas o secuencias típicas de eventos del guión.

En los guiones no existen reglas absolutas que definan contenidos genéricos. Así, un mismo evento puede ser contemplado según distintos factores de tiempo, lugares de ocurrencia o puntos de vista.

En los guiones el proceso de razonamiento está basado en que, si un determinado guión es apropiado para describir una situación dada, debe poder ser también utilizado para predecir acontecimientos no mencionados explícitamente. Así, para razonar sobre la base de un guión, éste debe ser previamente seleccionado mediante lo que se denomina proceso de *activación*.

Existen dos tipos fundamentales de guiones:

- ⋆ Los *guiones instantáneos* (aquellos a los que podemos referirnos siempre pero que no son el foco de atención principal de nuestro problema).
- ⋆ Los *guiones no instantáneos* (que constituyen el foco de atención principal de nuestro problema).

La activación de un guión instantáneo se realiza definiendo punteros en lugares estratégicos de forma que, cada vez que tal guión sea requerido, sepamos dónde buscarlo. Por el contrario, en el caso de los guiones no instantáneos, procede la activación total del mismo tras el proceso de emparejamiento de la situación inicial con el conjunto de guiones que describen secuencias en nuestro dominio. Así, la secuencia de eventos en un guión puede entenderse como una *cadena causal* gigante.

4.4. Paradigma de Orientación a Objetos

Desde mediados de la década de los 70 se produjo una fertilización recíproca entre la programación orientada a objetos y la investigación y el desarrollo en IA, lo que condujo a diversas y útiles extensiones de los lenguajes de IA.

Los *objetos* aparecen como una extensión del concepto de *frames* y su principal característica es que encapsulan en una misma representación, los datos y las estructuras procedimentales encargadas de manejar esos datos[5].

Aunque esta aproximación pueda parecer poco natural, ya que los sistemas de IA siempre han tendido a separar la información declarativa de la procedimental, se corresponde con una visión en la cual el mundo no está formado por datos y procedimientos, sino por entidades que encapsulan sus propios datos y los procedimientos para manejarlos, lo que provee de una base más firme para el desarrollo de sistemas.

Un **objeto** se define como una colección de información (atributos) que representa una entidad del mundo real y una descripción de cómo esa información es manipulada (métodos). La forma de comunicación entre objetos es mediante el envío de mensajes. Mensajes y métodos son dos caras de la misma moneda, los métodos son los procedimientos que se invocan cuando un objeto recibe un mensaje[6].

[5]Recordemos que en las frames la incorporación de información procedimental sólo aparecía en forma de *demons*.

[6]En terminología de programación tradicional, un mensaje es una llamada a una función.

Los objetos pertenecen a **clases**, en la práctica una *clase* es como un esquema o plantilla que se utiliza para definir o crear objetos.

Los cinco elementos más importantes del esquema de orientación a objetos son:

- Abstracción.
- Encapsulamiento.
- Modularidad.
- Jerarquía.
- Polimorfismo.

Los veremos en las secciones siguientes.

4.4.1. Abstracción

La **abstracción** consiste en ignorar aspectos de una entidad que no son relevantes para el problema actual, de forma que podamos centrar nuestra atención en aquellos aspectos que sí lo son. La aproximación de la O.O. fomenta que el desarrollador use abstracciones en los datos y en los procedimientos para simplificar su descripción del problema.

Definir una *abstracción* significa describir una entidad del mundo real, no importa lo compleja que pueda ser, y a continuación utilizar esa descripción en un programa. El elemento clave de la abstracción es la *clase*, que se puede definir como una descripción abstracta de un grupo de objetos, cada uno de los cuales se diferencia por su estado específico y por la posibilidad de realizar una serie de operaciones.

4.4.2. Encapsulamiento

El **encapsulamiento** se define como el proceso de almacenar en un mismo compartimento los elementos de una abstracción que constituyen su estructura y su comportamiento. *Abstracción* y *encapsulamiento* son conceptos complementarios: la primera se centra en el comportamiento observable de un objeto y el segundo en la implementación que da lugar a ese comportamiento.

El *encapsulamiento* también implica ocultación de información, de forma que cada objeto revela lo menos posible de su estructura interna. Esta parte pública del objeto es lo que se conoce como *interfaz*, mientras que los detalles internos del objeto que se ocultan al exterior se denominan *implementación*.

Así, una operación es vista por sus usuarios como si fuera una simple entidad, aunque en realidad esté formada por una secuencia de operaciones a bajo nivel. También un objeto es visto como un simple objeto en vez de como una composición de sus partes individuales. La supresión de los detalles de bajo nivel nos permite razonar acerca de la operación u objeto de forma más eficiente.

Un encapsulamiento inteligente permite que cambios en el diseño afecten al sistema sólo de forma local. Así, a medida que evoluciona un sistema, sus desarrolladores pueden ver que una aplicación real o bien ciertas operaciones llevan demasiado tiempo, o bien

ciertos objetos consumen un espacio excesivo. En estas situaciones se suele cambiar la representación de un objeto, pero sin alterar su interfaz. Esta capacidad para cambiar la representación de una abstracción sin alterar a ninguno de los clientes que la utilizan es una de las ventajas más importantes del encapsulamiento.

4.4.3. Modularidad

El hecho de fragmentar un programa en componentes individuales suele contribuir a reducir su complejidad en algún grado, además de crear una serie de fronteras bien definidas y documentadas dentro del programa, que tienen un valor muy alto de cara a la comprensión del mismo.

La **modularidad** es la propiedad que tiene un sistema que ha sido descompuesto en un conjunto de partes o módulos que sean *cohesivos* (agrupando abstracciones que guarden cierta relación lógica) y *débilmente acoplados* (minimizando las dependencias entre módulos).

Los principios de abstracción, encapsulamiento y modularidad son sinérgicos: un objeto proporciona una frontera bien definida alrededor de una sola abstracción, y tanto el encapsulamiento como la modularidad proporcionan barreras que rodean esa abstracción. En cualquier caso, encontrar las clases y los objetos correctos y organizarlos después en módulos separados son decisiones de diseño independientes. La identificación de clases y objetos es parte del diseño lógico de un sistema, mientras que la identificación de los módulos es parte del diseño físico del mismo. No pueden tomarse todas las decisiones de diseño lógico antes de tomar todas las decisiones de diseño físico y viceversa, sino que estas decisiones de diseño se dan de forma iterativa.

4.4.4. Jerarquía

Una **jerarquía** es una clasificación de las abstracciones. Como hemos visto anteriormente, las dos jerarquías más importantes en un sistema complejo son la *jerarquía de generalización/especialización*, que define relaciones ES_UN y la *jerarquía de agregación*, que define relaciones ES_PARTE_DE.

Las *jerarquías de generalización/especialización* también se conocen como *herencia*. Básicamente, la herencia define una relación entre clases, en las que una clase comparte la estructura de comportamiento definida en una o más clases (herencia simple y herencia múltiple, respectivamente). La herencia representa así una jerarquía de abstracciones, en las que una subclase hereda de una o más superclases. Típicamente, una subclase aumenta o redefine la estructura y el comportamiento de sus superclases.

A medida que se desarrolla la jerarquía de herencias, aquellas estructuras y comportamientos comunes a diferentes clases tenderán a migrar hacia superclases comunes. Las superclases representan abstracciones generalizadas y las subclases representan especializaciones en las que los campos y métodos de la superclase experimentan añadidos, modificaciones o incluso ocultaciones. Así, la herencia permite declarar las abstracciones con economía de expresión.

Si realizamos una pequeña reflexión sobre los conceptos que hemos visto hasta el momento, vemos que existe una cierta tensión entre encapsulamiento y jerarquía. El encapsulamiento intenta proporcionar una barrera opaca tras la que ocultar los métodos y el estado, mientras que la herencia requiere abrir esa interfaz en cierto grado y puede permitir el acceso a los métodos y al estado a sus subclases pero no a otro tipo de clases.

Esto responde a un principio fundamental de la técnica de descomposición modular que se conoce como **principio abierto-cerrado**. Este principio simplemente dice que los módulos deben ser a la vez abiertos y cerrados: un módulo debe ser abierto en el sentido de estar disponible para ser extendido a través de la herencia, y debe ser cerrado en el sentido de que su descripción o interfaz sea estable y esté bien definida. Es la forma de conjugar dos objetivos incompatibles: mantener los módulos abiertos a modificaciones posteriores y dar al sistema la estabilidad suficiente de forma que un cambio en un módulo no implique una reacción en cadena de cambios en muchos otros módulos que se basen directa o indirectamente en él[7].

4.4.5. Polimorfismo

Polimorfismo significa literalmente "capacidad para adoptar varias formas". En el desarrollo O.O. el *polimorfismo* puede aparecer de varias formas, siendo la más común de ellas aquélla en que a una variable de una superclase se le asigna un objeto perteneciente a una de sus subclases.

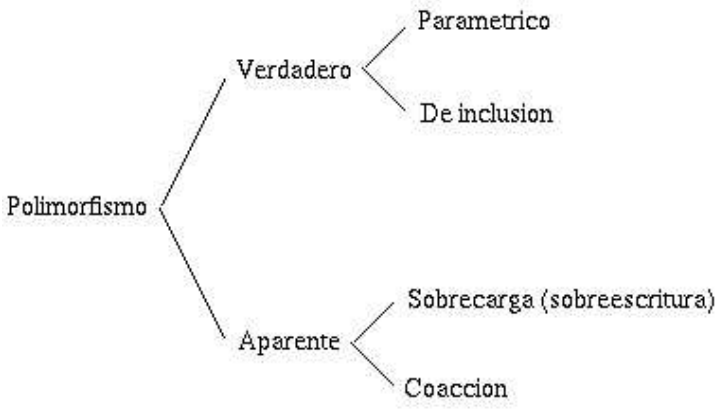


Figura 4.2: Tipos de *polimorfismo* en O.O.

El *polimorfismo* suele combinarse con otra característica, denominada *ligadura dinámica*, que permite decidir qué método se aplicará a un determinado objeto en tiempo de ejecución y no en tiempo de compilación.

Otra forma de polimorfismo es lo que se conoce como *sobrecarga*: decimos que un método está sobrecargado cuando a dicho nombre se le pueden asignar distintos cuerpos. La forma de determinar qué cuerpo del método hay que ejecutar en cada momento puede hacerse por los parámetros del método (sobrecarga paramétrica) o por la clase a la que pertenece el método (sobrecarga de mensajes).

[7]Relacionado con esto, en programación O.O. se manejan los conceptos de *atributo público*, *privado* o *protegido* y *métodos de acceso de escritura* y *de lectura*.

4.4.6. Ventajas e Inconvenientes de la O.O.

El paradigma de la O.O. presenta varias ventajas entre las que podemos citar:

- √ *Flexibilidad y extensibilidad*: la combinación de las características de herencia, polimorfismo, ligadura dinámica, etc. hacen posible utilizar y definir de forma clara módulos funcionalmente incompletos que permiten su extensión sin transtornar la operación de otros módulos o de sus clientes. De esta forma, los sistemas son flexibles, fácilmente extensibles y de mantenimiento menos costoso.
- √ *Reutilización*: los objetos bien diseñados constituyen la base para otros sistemas que se ensamblan en gran parte a partir de módulos reutilizables, lo que redunda en una mayor productividad.
- √ *Escalabilidad*: la combinación de la reutilización con la extensibilidad permite desarrollar nuevos sistemas complejos a partir de partes ya desarrolladas, lo que reduce la complejidad del desarrollo.
- √ *Naturalidad*: la aproximación O.O. corresponde a una visión más natural del mundo real que los típicos diseños *top-down* que se basan en una descomposición funcional por refinamiento progresivo.
- √ *Seguridad*: la ocultación de la información contribuye a la construcción de sistemas seguros.

Sin embargo, esta técnica de representación del conocimiento no está exenta de inconvenientes entre los que citamos:

- ∗ *Rendimiento*: el empleo profuso de la herencia, el polimorfismo y la ligadura dinámica ofrece mucha potencia al desarrollo pero su complejidad puede afectar al rendimiento global del sistema al consumir más recursos (memoria, tiempo,...).
- ∗ *Problemas de reutilización*: es fácil caer en diseños ad-hoc no pensados para su reutilización posterior. Un diseño reutilizable necesita una consideración especial y un coste de desarrollo mayor.
- ∗ *Cambio de cultura*: la aproximación a objetos obliga a pensar en términos de objetos y mensajes, mientras que la programación habitual está pensada en términos de datos y funciones. El cambio de puntos de vista no siempre es fácil.

4.5. Reglas de Producción

Las **reglas de producción** son esquemas de representación del conocimiento que pertenecen a lo que hemos denominado *métodos procedimentales* de representación, en los que la mayor parte de los conocimientos se representan como procedimientos dinámicos.

Estructuralmente, las **reglas de producción** son elementos de representación del conocimiento dinámico constituidas por:

- una *parte IF* (condición o premisa)
- una *parte THEN* (conclusión o acción)
- opcionalmente, una *parte ELSE* (conclusión-acción alternativa)[8]

La premisa de una regla está constituida por un conjunto de *cláusulas* que pueden anidarse a través de los juntores u operadores lógicos de relación AND, OR y el modificador lógico NOT.

Una vez definida la estructura clausal, el siguiente paso es encontrar una representación interna adecuada, tanto para las cláusulas como para las acciones y alternativas. Este proceso es dependiente de la herramienta utilizada y/o del lenguaje empleado, pero siempre debe ser compatible con la representación elegida para el conocimiento estático o declarativo, por lo que normalmente se realiza a través de ternas (parametro|relación|valor), donde los parámetros son las características que queremos investigar (deben compararse con los valores dados a través del operador de relación definido, para determinar si la cláusula es cierta o no).

En la tabla 4.2 (página 63) puede verse un ejemplo de cómo las reglas pueden cooperar con otras estructuras de representación declarativa del conocimiento, por ejemplo frames.

En algunos casos de anidamientos homogéneos de cláusulas, es útil definir y considerar distintos tipos de reglas:

√ Reglas *IFALL*, en las que todas las cláusulas de la premisa han de ser ciertas para que se ejecute la acción o se establezca la conclusión de la parte *THEN*; equivalen a una regla en la que todas las cláusulas estén anidadas por medio de operadores AND.

√ Reglas *IFANY*, en las que al menos una cláusula de la premisa ha de ser cierta para que se ejecute la acción o se establezca la conclusión; equivalen a una regla en la que todas las cláusulas estén anidadas por medio de operadores OR.

√ Reglas *IFSOME*, en las que al menos una cláusula de la premisa ha de ser cierta para que se ejecute la acción o se establezca la conclusión; equivalen a una regla en la que todas las cláusulas estén anidadas por medio de operadores OR.

[8] En contra de lo que pudiésemos pensar, la acción *else* no se ejecuta sólo cuando la premisa es falsa, sino también en el caso de que no se tenga información suficiente como para determinar su veracidad, lo que nos permite, pues, el trabajo con conocimiento incompleto.

```
(HEMODYN-1) /* regla 1 */
IF   : (presion_arterial sistolica) gt 160
AND  : (presion_arterial diastolica) gt 95
AND  : (presion_arterial media) gt 130
THEN : (diagnostico hemodinamico hipertension_arterial)
AND  : (ACTUALIZAR (diagnostico hemodinamico hipertension_arterial)
                   base_de_datos
       )

(GASOM-1) /* regla 2 */
IF   : (gases_arteriales CO2) eq hypercapnia
AND  : (gases_arteriales pH) eq acidemia
AND  : (gases_arteriales Bic) eq normal
THEN : (diagnostico respiratorio acidosis_respiratoria)

/* frame 1 */

presion_arterial
   sistolica
      177
   diastolica
      99
   media
      131

/* frame 2 */

gases_arteriales
   CO2
      hypercapnia
   pH
      acidemia
   Bic
      normal

/* nueva frame generada tras la ejecucion */

diagnostico
   hemodinamico
      hipertension_arterial
   respiratorio
      acidosis_respiratoria
```

Cuadro 4.2: Ejemplo de funcionamiento de reglas de producción y frames.

La diferencia entre las reglas *IFANY* y/o *IFSOME* es que en las primeras cuando se encuentra una cláusula que verifica la condición ya no es preciso investigar más y la acción se ejecuta o la hipótesis se establece, mientras que en las segundas se investigan todas las cláusulas antes de ejecutar la acción o establecer la hipótesis (lo que se hace igualmente cuando al menos una de ellas es cierta). Es decir, una regla *IFANY* formaliza una búsqueda no exhaustiva y una *IFSOME* representa una búsqueda exhaustiva (que, por consiguiente, nos hará conocer más cosas del dominio). Asimismo, la estructura de los hechos representados por las reglas *IFANY* e *IFSOME* cambia también, pues aquéllas consideran un OR exclusivo y éstas implican un OR inclusivo.

La definición de distintos tipos de reglas no excluye en absoluto la posibilidad de nuevos anidamientos, aunque se considera una mala práctica (es mejor desdoblar las veces que haga falta y obtener una estructura de conocimiento más compacta), es decir, a pesar de que sería válida una regla:

```
IFALL : OR :
             A verdadero
             NOT C=desconocido
        NOT V=blanco
        Luis=alto
THEN  : H=hipótesis
```

es preferible esta otra opción:

```
IFALL : A verdadero
        NOT V=blanco
        Luis=alto
THEN  : H=hipotesis

IFALL : NOT C=desconocido
        NOT V=blanco
        Luis=alto
THEN  : H=hipotesis
```

La representación del conocimiento mediante reglas de producción presenta ciertas ventajas:

- Las condiciones y las acciones involucradas son explícitas.
- El conocimiento es expresado de forma muy modular, ya que cada regla de un sistema dado constituye una unidad completa de conocimiento (y, por tanto, si es eliminada, el sistema sigue funcionando).
- Las reglas de producción permiten almacenar y utilizar conocimiento de una gran especifidad y el conocimiento implicado suele ser de naturaleza heurística.

Una búsqueda dirigida por lo objetivos utilizando reglas de producción se haría entrando por la parte *THEN*, mientras que una búsqueda dirigida por los datos, se haría desde las condiciones hacia las conclusiones.

Capítulo 5

Sistemas de Producción

Los **sistemas de producción** pueden definirse como sistemas inteligentes basados en *reglas* (de producción) en los que los *mecanismos de emparejamiento* son una *parte explícita* de su arquitectura (estructura)[1].

Podemos clasificarlos en dos categorías, según la sintaxis de sus reglas y su estructura de control:

- *Sistemas de producción dirigidos por los datos, por los antecedentes, o progresivos*: en ellos las inferencias se obtienen cuando los antecedentes de una o más de sus reglas de producción se empareja con al menos una parte de los hechos que describen el estado actual. No todos los hechos tienen por qué verse representados en la regla, pero el antecedente de ésta sí debe estar completamente representado en los hechos. Cuando esto ocurre, se dice que la regla en cuestión se ha *activado* y está en condiciones de ser *ejecutada*, o, lo que es lo mismo, se ha seleccionado como potencialmente ejecutable (su ejecución o no dependerá de la estrategia de exploración elegida).
- *Sistemas de producción dirigidos por los objetivos, por las metas, o regresivos, evocativos*: en ellos tanto los antecedentes como los consecuentes de las reglas deben ser considerados como *aserciones* sobre los datos. En este caso, la activación de las reglas tiene lugar por medio de un encadenamiento regresivo (hacia atrás), y el emparejamiento se efectúa a través de las conclusiones de las reglas. Así, para alcanzar una determinada meta hay que configurar un proceso evocativo en el que de forma recursiva se vayan estableciendo los antecedentes de las metas como submetas de orden inferior.

Un sistema de producción está constituido por tres elementos fundamentales:

√ La base de conocimientos

√ La memoria activa

√ El motor de inferencias

[1] Esto hace que sean lentos, pero también muy informativos.

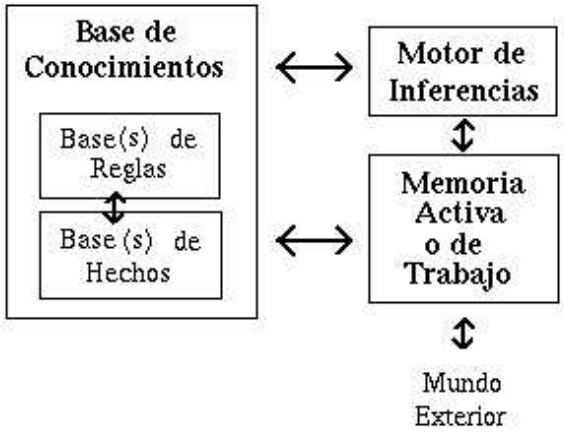


Figura 5.1: Arquitectura básica de un sistema de producción.

5.1. Base de Conocimientos

La **base de conocimientos** describe el universo de discurso o dominio en el que el sistema de producción tiene que plantear soluciones. Está constituida por *bases de hechos* y *bases de reglas* (relativas a diferentes aspectos del dominio[2]).

Las bases de hechos forman el esqueleto declarativo del sistema de producción y su misión es articular estáticamente todos los hechos potencialmente relevantes del dominio[3] (esto es, los datos o hechos –inferenciales o no– que forman parte de las reglas). Puede decirse que las bases de hechos dan una estructura a las bases de reglas.

Por su parte, las bases de reglas constituyen el esqueleto procedimental del sistema de producción y a través de ellas se posibilita la construcción de los *circuitos inferenciales* que permitirán obtener conclusiones válidas.

Obviamente la estructura de las bases de hechos y las bases de reglas debe ser "compatible", es decir, tal que ambas entidades puedan "comprenderse" entre sí.

5.2. Memoria Activa

La **memoria activa** o **memoria de trabajo** es la estructura que contiene toda la información de naturaleza estática necesaria para resolver (plantear) un problema concreto. Esta información incluye:

- datos iniciales del problema (información de partida)
- datos incorporados con posterioridad

[2]Por ejemplo, podrían agruparse todas las entidades relevantes para el estudio de la química en una base de hechos, y el conjunto de todas las relaciones que se pueden establecer entre dichas entidades en una base de reglas. Ambas estructuras podrían estar diferenciadas respecto al conocimiento relevante –*química orgánica*, *inorgánica*, *analítica*,...–.

[3]Un ejemplo serían conjuntos de frames.

- hechos establecidos durante los procesos inferenciales
- hipótesis de trabajo, metas o submetas que todavía no han sido establecidas[4]

En la memoria activa es donde se producen todos los cambios de estado del sistema, de forma que es la memoria activa la que representa siempre el estado actual. Por esta razón, es la responsable de interaccionar con el mundo exterior, aceptando la entrada de información de naturaleza no inferencial, y es también el foco permanente de atención de las reglas del sistema. En ella, mediante procesos inferenciales, se activarán y ejecutarán reglas que harán desaparecer submetas como hipótesis al confirmarlas como hechos (ambas representaciones no pueden, lógicamente, coexistir). La potencia de este mecanismo se refleja también en que, paralelamente, surge información, como resultado de los emparejamientos con las reglas, que no se buscaba.

Cuando el proceso se detiene, la memoria activa contiene una descripción del estado final del problema, datos, hechos e hipótesis (alternativas o proporcionadas).

Los hechos y los datos de la memoria activa corresponden a entidades de la base de hechos, pero con valores concretos asociados. La diferencia entre hechos y datos es la procedencia de dichos valores asociados: mientras los valores de las trayectorias correspondientes a los hechos de la base son asignados a través de un proceso inferencial (procedencia interna), los datos representan información que procede directamente del mundo exterior. Por último, las hipótesis son trayectorias completas (incluyendo –o no– valores), cuya veracidad se desea investigar.

5.3. Motor de Inferencias

El **motor de inferencias** consta de dos entidades: un *intérprete* y una *estrategia de control* (global, que puede estar constituida por varias estrategias de control concretas), físicamente separados de la base de conocimientos del dominio de aplicación. Contiene los mecanismos necesarios para[5]:

- Examinar la memoria activa y determinar qué reglas deben ejecutarse, en función de la estrategia de búsqueda elegida y de los modelos implementados para la resolución de los conflictos que pudiesen aparecer (estrategia de control).
- Controlar y organizar el proceso de ejecución de las reglas seleccionadas.
- Actualizar la memoria activa.
- Asegurar que el sistema tiene autoconocimiento (saber qué reglas han sido activadas en todo momento, cuáles han sido ejecutadas, qué hecho ha sido el último en ser incorporado, etc.).

[4]Por ejemplo, en sistemas de producción dirigidos por los objetivos. Se indican mostrándolas entre paréntesis.

[5]Véase tema 2, página 17.

El proceso global de trabajo del motor de inferencias se produce por ciclos denominados *ciclos básicos del sistema de producción*, cuya naturaleza es netamente diferente si el proceso de búsqueda está dirigido por los datos o lo está por los objetivos.

En cualquier caso, todo motor de inferencias debe ser considerado como un intérprete y, como tal, no es más que un programa de naturaleza secuencial cuya misión es decidir qué es lo que hay que hacer en cada momento, reconociendo y activando las reglas apropiadas en función de[6]:

- √ los criterios de activación elegidos
- √ las estrategias de búsqueda implementadas
- √ la dirección de tránsito por el espacio de estados
- √ el test de realización (prueba de meta)

Además, para tratar de optimizar el proceso de exploración del espacio de estados, la estrategia de control debe observar criterios como la producción de movimientos válidos en el susodicho espacio de estados, sistematicidad y eficiencia[7].

Así pues, el motor de inferencias es quien gobierna los procesos inferenciales en los sistemas de producción y, dado que éstos están basados esencialmente en reglas, consideraremos, desde una perspectiva totalmente general, dos procedimientos básicos de materialización de la propagación del conocimiento en el sistema:

- ◦ Encadenamiento progresivo (proceso de búsqueda dirigido por los datos)
- ◦ Encademaniento regresivo (proceso dirigido por los objetivos)

Además del tipo de encadenamiento de reglas, todo motor de inferencias debe incluir, como norma general:

- ▶ emparejador o intérprete, que active las reglas relevantes en cada momento de acuerdo con el estado de la memoria activa
- ▶ estrategia de búsqueda, que incluya heurísticas de exploración del espacio de estados
- ▶ mecanismos de autoconocimiento, que permitan identificar estructuras utilizadas, estados del problema, cambios en la memoria activa, órdenes de prioridades de aciones y hechos inferidos,...
- ▶ mecanismos de terminación de los procesos inferenciales

[6] Grandes casas comerciales, como IBM desarrollan productos comerciales especializados de este tipo.
[7] Ver ejemplos en [1], capítulo 6, sección 6.4, página 147 y siguientes.

5.4. Ciclo básico de un Sistema de Producción

Independientemente de si un proceso inferencial está dirigido por los antecedentes o por las metas, el ciclo básico de un sistema de producción está constituido por dos fases claramente diferenciadas:

1. Fase de decisión o selección de reglas, que involucra:
   a) Restricción.
   b) Equiparación.
   c) Resolución de conflictos.
2. Fase de acción o ejecución de las reglas seleccionadas.

La *restricción* trata de simplificar el proceso de *equiparación* eliminando del foco de atención del motor de inferencias aquellas reglas que claramente no tienen nada que ver con el estado actual representado en la memoria activa del sistema. Esta tarea suele ser realizada durante la fase de *ingeniería del conocimiento* o fase de diseño y construcción del sistema inteligente, y típicamente se traduce en una organización del dominio de discurso por "temas". Esta forma de abordar la etapa de restricción es llevada a cabo a priori y es de marcada naturaleza estática. Existe una alternativa de naturaleza más dinámica, que consiste en abordar la restricción a partir del llamado *metaconocimiento* (conocimiento sobre conocimiento), formalizado como *metarreglas* que a nivel local y dentro de un proceso inferencial dado, son capaces de establecer prioridades a la hora de acometer el proceso de activación de reglas. Las metarreglas también pueden utilizarse en la fase de acción durante la etapa de resolución de conflictos.

La *equiparación* (o *emparejamiento*) trata de identificar qué reglas son potencialmente relevantes en el contexto del problema que queremos resolver (conjunto conflicto). Si el proceso sigue un encadenamiento progresivo (dirigido por los datos), consistirá en seleccionar aquellas reglas cuyos antecedentes estén representados en hechos y/o datos de la memoria activa. Si por el contrario el encadenamiento es regresivo, se encontrarán aquellas reglas que concluyan algo sobre hipótesis presentes en la memoria activa.

Finalmente, en la etapa de *resolución de conflictos* se decidirá qué regla (o reglas) se aplican. Esta decisión está fuertemente condicionada por la estrategia genérica de búsqueda implementada en la estructura de control del motor de inferencias.

Una vez superada la fase de decisión o selección de reglas, el sistema está preparado para "dispararlas" físicamente. Es la fase de *acción* o *ejecución*, cuyo resultado es la actualización de la memoria activa con nuevos hechos y/o hipótesis, el *marcaje* de las estructuras utilizadas y la verificación de si el proceso cíclico debe continuar o ha finalizado (se ha encontrado la solución al problema o bien no pueden utilizarse más reglas).

Capítulo 6

Representación de Conocimiento Temporal

En todo sistema físico en estudio, la inclusión de la variable *tiempo* incrementa la dificultad asociada al tratamiento de los problemas del dominio. La IA contempla el problema temporal desde dos puntos de vista: la *representación* computacional de la información dependiente del tiempo y el *razonamiento* basado en información temporal.

La utilización de información temporal es importante en muchos dominios y problemas, ya que permite identificar contextos temporales, comparar datos pasados entre sí, realizar predicciónes, etc.

El problema de la representación del conocimiento temporal puede ser tratado sintáctica o semánticamente. El tratamiento sintáctico se puede realizar mediante asociaciones etiqueta-evento o mediante grafos de evolución en los que se indiquen relaciones temporales y no sólo relaciones causales; desde una perspectiva semántica, el tiempo suele emplearse como contexto y estar implícito.

En relación con los procesos inferenciales el tiempo es importante porque permite el establecimiento de relaciones causales, importantísimas en IA, resultado de considerar conjuntamente hallazgos efectuados durante el proceso inferencial y la cronología de los mismos. El establecimiento de estas relaciones causales aumenta las capacidades predictivas de los sistemas inteligentes.

Sea como fuere, los problemas clave de la representación del conocimiento temporal son dos:

- $\surd$ representación del eje temporal
- $\surd$ ajuste de la granularidad

Podemos considerar que el eje temporal está constituido por una secuencia de puntos discretos, de forma que los eventos suceden en instantes concretos de dicho eje, o bien podemos suponer que el eje temporal es una secuencia continua de intervalos, de forma que los eventos suceden en alguno de tales segmentos temporales. En ambos casos, la representación del conocimiento será diferente.

En cuanto al ajuste de granularidad, es importante que la representación sea capaz de contemplar intervalos de tiempo más o menos largos dependiendo del contexto.

Claro está que, además de considerar los puntos anteriores, es necesario un modelo capaz de manejar la información temporal. Veremos algunos en el presente capítulo.

6.1. Especialista Temporal de Kahn y Gorry

La aproximación de Kahn y Gorry (1977) es uno de los primeros trabajos que abordan la problemática de la representación temporal. Afirman que el conocimiento temporal puede incluirse en gran parte en un conjunto de rutinas a las que se refieren colectivamente como "el especialista temporal", que estaría al servicio de los programas de resolución de problemas para tratar aquellas cuestiones temporales que apareciesen en el dominio. Este *especialista* se alimenta con una serie de sentencias que hacen referencia a cuestiones temporales; el programa puede instar al especialista a realizar deducciones y responder a preguntas sobre dichas sentencias temporales. Además, éste acepta especificaciones temporales en diversos formatos, detecta inconsistencias y vuelve a deducir hechos basados en las especificaciones que causaban la inconsistencia.

6.1.1. Representación de las referencias temporales

Una **especificación temporal** es una sentencia que parcialmente establece la relación temporal entre dos eventos, cada uno de los cuales puede ser considerado un punto de tiempo. Las especificaciones temporales que referencian más de dos eventos pueden ser descompuestas en sentencias más sencillas que sólo involucren dos eventos temporales. Uno de los eventos de la especificación suele actuar como evento de referencia.

El hecho de utilizar puntos de tiempo obliga a que las referencias temporales que involucran intervalos temporales sean divididas en dos eventos separados que corresponden al inicio y al final de la ocurrencia.

6.1.2. Organización de las especificaciones temporales

El modo según el que el especialista temporal organiza las especificaciones temporales es importante, ya que esta organización tiene mucha influencia en la eficiencia con que se responde a las distintas cuestiones. Puede hacerlo de tres maneras[1]:

1. Mediante *fechas*, insertando los eventos en una línea temporal según su fecha, admitiendo expresiones difusas permitiendo incluir sus límites superior e inferior.
2. Mediante *eventos de referencia*, cuando hay eventos que son usados con frecuencia, y cuya fecha se conoce con exactitud, pueden ser usados para calcular la fecha de otros eventos relacionados con ellos.
3. Mediante *cadenas antes/después*, que ocurren cuando los eventos principales forman una secuencia.

[1] La elección depende del usuario

6.1.3. Preguntas al especialista temporal

El especialista temporal puede responder a tres tipos de preguntas acerca de los hechos almacenados en sus bases de datos:

1. *¿Sucedió X en la expresión temporal T?*
2. *¿Cuándo sucedió X?*
3. *¿Qué sucedió en la expresión temporal T?*

La capacidad del especialista temporal de responder preguntas reside en un conjunto de programas llamados colectivamente *fetcher*, cuyas tareas son aceptar un patrón que especifica una pregunta, interpretarlo para determinar el tipo de pregunta y seleccionar los métodos adecuados para responderla. Cada método es un programa independiente diseñado para responder un tipo particular de pregunta haciendo uso de la organización de hechos concreta de la base de datos.

6.2. Modelo de Allen

Este segundo modelo se basa en la utilización de *intervalos* de tiempo como elementos fundamentales para establecer relaciones temporables, justificándolo alegando que su utilización permite representar de forma más natural la información temporal, ya que, por ejemplo:

- Normalmente las referencias temporales son vagas e implícitas, más fáciles de representar con intervalos que con puntos.
- Algunos eventos parecer ser instantáneos pero si los examinamos minuciosamente veremos que pueden ser descompuestos en nuevos eventos, que a su vez se podrían descomponer en otros nuevos, razón por la cual la utilización de puntos de tiempo no sería útil ya que éstos no se pueden descomponer.
- Existen ejemplos en los que el uso de puntos de tiempo (de "anchura" cero) nos conducen a situaciones problemáticas (cambio de encendido a apagado en una bombilla, fácilmente asumible con intervalos cerrados en el inicio y abiertos en el final). Ahora bien, si permitimos puntos de tiempo, los intervalos pueden ser representados por sus puntos finales pudiendo definir un intervalo como un par ordenado de puntos sobre la línea real que define el tiempo, donde el primer punto es menor que el segundo. Esta solución, no obstante, no es conveniente según Allen debido a que no facilita estructurar el conocimiento de una manera adecuada para la realización de tareas típicas de razonamiento temporal.

6.2.1. Relaciones Temporales de Allen

Una vez establecido que el elemento básico de la representación temporal es el intervalo, es necesario definir las posibles relaciones existentes entre dichos intervalos. Allen define

un total de trece posibles relaciones entre un par ordenado de intervalos de tiempo (tabla 6.1, página 76).

Relacion	**Simbolo**	**Simbolo para la inversa**	**Ejemplo**
X antes Y (before)	`< / b`	`> / bi`	X Y
X igual Y (equal)	`= / eq`	`sin inversa`	X Y
X seguido de Y (meets)	`m`	`mi`	X Y
X superpuesto a Y (overlaps)	`o`	`oi`	X Y
X durante Y (during)	`d`	`di`	X Y
X comienza Y (starts)	`s`	`si`	X Y
X finaliza Y (finishes)	`f`	`fi`	X Y

Figura 6.1: Las 13 relaciones temporales de Allen.

Las relaciones entre intervalos se representan en una red donde los nodos son intervalos individuales y los arcos entre dichos nodos llevan etiquetas que indican las posibles relaciones existentes entre ellos. Si existe incertidumbre sobre la relación que debe existir en un determinado arco, la solución propuesta es poner todos los casos posibles en el arco.

La red mantiene siempre una información completa sobre los intervalos. Cuando se introduce una nueva relación se deben calcular todas las consecuencias que conlleve. Esto se hace calculando el *cierre transitivo* de las relaciones temporales: el nuevo hecho añade una restricción sobre cómo sus dos intervalos deberían ser relacionados, lo que podría, sucesivamente, introducir nuevas restricciones entre nuevos intervalos a través de las reglas de transitividad que gobiernan las relaciones temporales, que se muestran en la tabla 6.1[2].

El algoritmo para realizar la propagación transitiva consta de una subrutina llamada `Restricciones` que es la función de transitividad en sí para listas de relaciones (etiquetas de los arcos); se asume la existencia de una cola denominada `PorHacer` que almacena las relaciones a procesar, dos intervalos i y j relacionados por el conjunto de relaciones $N(i, j)$

[2]Esta tabla omite la relación = y muestra las relaciones posibles entre A y C dadas las relaciones entre A-B y B-C.

B r2 C / A r1 B	<	>	d	di	o	oi	m	mi	s	si	f	fi
<[a]	<	??	< o m d s	<	<	< o m d s	<	< o m d s	<	<	< o m d s	<
>[b]	??	>	> oi mi d f	>	> oi mi d f	>	> oi mi d f	>	> oi mi d f	>	>	>
d[c]	<	>	d	??	< o m d s	> oi mi d f	<	>	d	> oi mi d f	d	< o m d s
di[d]	< o m di fi	> oi di mi si	o oi d di =	di	o di fi	oi di si	o di fi	oi di si	di fi o	di	di si oi	di
o[e]	<	> oi di mi si	o d s	< o m di fi	< o m	o oi d di =	<	oi di si	o	di fi o	d s o	< o m
oi[f]	< o m di fi	>	oi d f	> oi mi di si	o oi d di =	> oi mi	o di fi	>	oi d f	oi > mi	oi	oi di si
m[g]	<	> oi mi di si	o d s	<	<	o d s	<	f fi =	m	m	d s o	<
mi[h]	< o m di fi	>	oi d f	>	oi d f	>	s si =	>	d f oi	>	mi	mi
s[i]	<	>	d	< o m di fi	< o m	oi d f	<	mi	s	s si =	d	< m o
si[j]	< o m di fi	>	oi d f	di	o di fi	oi	o di fi	mi	s si =	si	oi	di
f[k]	<	>	d	> oi mi di si	o d s	> oi mi	m	>	d	> oi mi	f	f fi =
fi[l]	<	> oi mi di si	o d s	di	o	oi di si	m	si oi di	o	di	f fi =	fi

[a]Relación “antes”.
[b]Relación “después”.
[c]Relación “durante”.
[d]Relación “contiene”.
[e]Relación “superpuesto a”.
[f]Relación “superpone a”.
[g]Relación “seguido de”.
[h]Relación “sigue a”.
[i]Relación “comienza”.
[j]Relación “comenzado por”.
[k]Relación “finaliza”.
[l]Relación “finalizado por”.

Cuadro 6.1: Tabla de transitividad para las relaciones temporales.

y $R(i, j)$ como la nueva relación a añadir a la red temporal.

```
Restricciones (R1, R2)
{
  C = O
  para cada r1 en R1
    para cada r2 en R2
      C = C U T(r1, r2);

  return C
}

Añadir R(i,j)
{
  Añadir <i,j> a la cola PorHacer
  mientras (PorHacer /= vacía) hacer
  {
    coger el siguiente <i,j> de la cola PorHacer
    N(i,j) = R(i,j)
    para cada nodo k tal que Comparable(k,j) hacer
    {
      R(k,j) = N(k,j) ^ Restricciones(N(i,k), R(i,j))
      si R(k,j) = O entonces
        return contradicción
      si R(k,j) /= N(k,j) entonces
        añadir <k,j> a PorHacer
    }
    para cada nodo k tal que Comparable(i,k) hacer
    {
      R(i,k) = N(i,k) ^ Restricciones(R(i,j), N(k,j))
      si R(i,k) = O entonces
        return contradicción
      si R(i,k) /= N(i,k) entonces
        añadir <i,k> a PorHacer
    }
  }
}
```

Por sencillez, el predicado `Comparable` puede suponerse siempre cierto inicialmente.

Para reducir los requisitos de espacio de la representación sin afectar de forma importante a los mecanismos inferenciales, Allen introduce los **intervalos de referencia**, intervalos temporales que agrupan otros intervalos. Las restricciones temporales entre cada par de intervalos incluidos en un intervalo de referencia están calculadas de antemano.

Cada intervalo se relaciona con el resto de intervalos del sistema únicamente a través del intervalo de referencia, dando lugar a una jerarquía en árbol basada en intervalos de referencia.

En este caso es necesario redefinir el predicado `Comparable` de modo que para cualesquiera nodos K y J, `Comparable(k,j)` es cierto si:

- `Referencia(k)` $\cup$ `Referencia(j) /= 0` (comparten algún intervalo de referencia),
- `k` $\in$ `Referencia(j)`, o
- `j` $\in$ `Referencia(k)`.

donde `Referencia(n)` es el conjunto de intervalos de referencia para cualquier nodo N. La filosofía es que no se calculan las relaciones entre intervalos de distintos intervalos de referencia: si no hay relación directa entre dos nodos, se obtiene a través de la red por transitividad.

Ya que los intervalos de referencia tienen el mismo comportamiento que los intervalos temporales, pueden contener así mismo, como decimos, intervalos de referencia, definiendo de esta manera una jerarquía de grupos representable gráficamente mediante un árbol.

6.2.2. Lógica temporal de Allen

Basándose en las relaciones entre intervalos temporales descritas en el apartado anterior, Allen establece una lógica temporal que, básicamente, consiste en una extensión temporal de la lógica de predicados de primer orden.

La lógica consta de tres tipos básicos de términos:

- términos del tipo **intervalo de tiempo**, referidos a intervalos temporales
- términos del tipo **propiedad**, referidos a proposiciones (ciertas o no) durante un intervalo temporal
- términos correspondientes a objetos del dominio

Uno de los predicados más importantes es SE_MANTIENE(p, t)[3]. Otras primitivas básicas de relaciones entre intervalos se deducen de lo visto hasta ahora:

* DURANTE(t_1, t_2), t_1 está totalmente contenido en t_2
* COMIENZA(t_1, t_2), empiezan juntos pero acaba antes t_1
* FINALIZA(t_1, t_2), acaban juntos pero empieza antes t_2
* ANTES(t_1, t_2), t_1 es antes de t_2 y no coinciden
* SUPERPUESTO(t_1, t_2), t_1 empieza antes que t_2 y se solapan
* SEGUIDO(t_1, t_2), t_1 está justo antes que t_2

[3]En inglés, HOLDS(p, t), la propiedad p se cumple durante todo el intervalo t.

- IGUAL(t_1, t_2), son el mismo intervalo

De nuevo, son aplicables las reglas de la tabla de transitividad 6.1 y además el uso de AND, OR, ALL y EXISTS que permiten las expresiones lógicas permite construir otros predicados.

Como puede verse, el modelo de las relaciones temporales de Allen da a la variable temporal una representación formal de implementación sencilla gracias a su base en la lógica de predicados de primer orden.

6.2.3. Críticas al modelo de Allen

Las críticas al modelo de Allen se han centrado sobre todo en el hecho de no utilizar puntos de tiempo y basar su representación en intervalos temporales. Pueden clasificarse en dos vertientes:

- *Críticas a la representación del conocimiento*, entre las que destaca el trabajo de Galton (1990) quien argumenta que el modelo de Allen no es adecuado para representar hechos que están en movimiento continuo, que estarían en una posición diferente en cada instante de tiempo (y por tanto no pueden estar en una posición en un intervalo de tiempo, ya que eso significaría que durante ese intervalo estuvieron parados). Galton propone una revisión al modelo de Allen que incluya **instantes temporales** además de intervalos, añadiendo dos nuevos predicados:
  - DENTRODE(i, t), el instante i cae en el intervalo t
  - LIMITA(i, t), el instante i limita al intervalo t
- *Críticas a la complejidad de los cálculos* realizados con intervalos temporales, entre las que destacan los trabajos de Vilain y Kautz (1986) y van Beek (1992, 1996). Estos autores reconocen la sencillez y fácil implementación del modelo de Allen, pero indican que el álgebra de intervalos en la que se basa requiere gran cantidad de recursos computacionales, en particular, dos de las operaciones fundamentales:
  - la búsqueda de todas las relaciones posibles entre pares de intervalos (o puntos), que se hace mediante un razonamiento deductivo atendiendo a las relaciones transitivas entre intervalos temporales
  - la búsqueda de un escenario consistente con la información suministrada, lo que significa encontrar una subred de la red actual en la que cada nodo se etiquete con una sola relación y que exista una instanciación consistente de dicha subred (si no existe, la que no es consistente es la propia red)

  En el álgebra de intervalos estas dos operaciones son NP-completas. Para solucionar este problema, Vilain y Kautz, y posteriormente van Beek, proponen un álgebra más sencilla basada en puntos de tiempo y presentan un método para convertir las representaciones en intervalos temporales en representaciones en puntos temporales.

6.3. Álgebra de Puntos Temporales

Una alternativa al razonamiento basado en intervalos de tiempo es el basado en puntos de tiempo. En el **álgebra de puntos** éstos se relacionan entre sí a través de vectores de relación, cada uno de los cuales está compuesto por un conjunto de relaciones básicas entre puntos, que son las 3 que se indican en la figura 6.2, frente a las 13 del modelo de Allen:

Relacion	**Simbolo**	**Ejemplo**
X precede Y (precedes)	<	
X igual Y (same)	=	
X sigue Y (follows)	>	

Figura 6.2: Las tres posibles relaciones entre puntos de tiempo.

Vilain y Kautz definen además dos operaciones básicas entre puntos o intervalos de tiempo:

- **Suma**, intersección entre dos vectores que definen una relación entre intervalos o puntos para devolver el vector representante de la relación menos restrictiva permitida.
- **Multiplicación**, equivalente a la operación que el modelo de Allen denominaba `Restricciones`, que dados tres intervalos A, B y C y dos vectores que relacionen $A - B$ y $B - C$, permite obtener el vector menos restrictivo que relaciona $A - C$.

El álgebra temporal basada en puntos posee también las operaciones de adición y producto, cuyas tablas se muestran en la página 82.

Para las redes de puntos, van Beek presenta algoritmos más eficientes para realizar las tareas del razonamiento temporal. En concreto define un algoritmo de complejidad $O(n^2)$ en el tiempo para la tarea de encontrar un escenario consistente (donde n es el número de puntos) y uno de complejidad $O(max(mn^2, n^3))$ en el tiempo para encontrar todas las posibles relaciones entre puntos de tiempo (donde n es el número de puntos y m es el número de pares de puntos que no pueden considerarse iguales).

6.3.1. Álgebra de Puntos vs. Álgebra de Intervalos

La facilidad en el tratamiento del álgebra de puntos parece hacerla más adecuada para la representación del tiempo. La mayoría de los problemas que requieren manejar

	$<$	$\leq$	$>$	$\geq$	$=$	$\cong$	$?$
$<$	$<$	$<$	$\varnothing$	$\varnothing$	$\varnothing$	$<$	$<$
$\leq$	$<$	$\leq$	$\varnothing$	$=$	$=$	$<$	$\leq$
$>$	$\varnothing$	$\varnothing$	$>$	$>$	$\varnothing$	$>$	$>$
$\geq$	$\varnothing$	$=$	$>$	$\geq$	$=$	$>$	$\geq$
$=$	$\varnothing$	$=$	$\varnothing$	$=$	$=$	$\varnothing$	$=$
$\cong$	$<$	$<$	$>$	$>$	$\varnothing$	$\neq$	$\neq$
$?$	$<$	$\leq$	$>$	$\geq$	$=$	$\neq$	$?$

Cuadro 6.2: Adición en el álgebra de puntos temporales.

	$<$	$\leq$	$>$	$\geq$	$=$	$\cong$	$?$
$<$	$<$	$<$	$?$	$?$	$<$	$?$	$?$
$\leq$	$<$	$\leq$	$?$	$?$	$\leq$	$?$	$?$
$>$	$?$	$?$	$>$	$>$	$>$	$?$	$?$
$\geq$	$?$	$?$	$>$	$\geq$	$\geq$	$?$	$?$
$=$	$<$	$\leq$	$>$	$\geq$	$=$	$\neq$	$?$
$\cong$	$?$	$?$	$?$	$?$	$\neq$	$?$	$?$
$?$	$?$	$?$	$?$	$?$	$?$	$?$	$?$

Cuadro 6.3: Multiplicación en el álgebra de puntos temporales.

relaciones temporales pueden ser representados mediante puntos de tiempo.

Sin embargo, los puntos de tiempo son inadecuados para representar, por sí mismos, toda la semántica del lenguaje natural. Además, presenta muchos inconvenientes para modelizar muchos de los eventos y acciones del mundo real. En estos casos, la representación temporal basada en intervalos es mejor.

La mayoría de las relaciones basadas en intervalos tienen una traducción directa en álgebra de puntos: se puede considerar que un intervalo queda delimitado por sus puntos extremos (inicial y final), y traducir las relaciones existentes entre intervalos a relaciones entre los puntos extremos de dichos intervalos.

No obstante, no todas las relaciones entre intervalos pueden ser expresadas *sin pérdidas* como relaciones entre sus puntos de inicio y fin. En concreto lo que no puede expresarse con puntos son expresiones en las que aparecen disyunciones.

De esta forma, existen dos tipos de redes temporales: basadas en intervalos (*Interval Algebra*) y basadas en puntos (*Point Algebra*). También existe un subconjunto de las redes basadas en intervalos denominadas redes SIA (*Simple Interval Algebra*) que son directamente representables en base a redes basadas en puntos. Estas redes SIA incluyen todas las relaciones no ambiguas entre intervalos (de forma que no tienen pérdida de información), es decir, relaciones que pueden ser expresadas usando vectores que contienen sólo un simple constituyente (un sólo elemento). Incluso se consideran algunas relaciones ambiguas, pero no todas: se puede representar la ambigüedad en relación de pares de puntos finales, pero no en relación a intervalos completos.

Capítulo 7

Razonamiento Categórico y Corrección Bayesiana

La IA no sólo se ocupa de mecanismos generales relacionados con la búsqueda de soluciones en un espacio dado, o de cómo representar y utilizar el conocimiento de un determinado dominio de discurso. Otro aspecto es el que corresponde a los mecanismos y/o procesos inferenciales, punto de partida de los llamados *modelos de razonamiento*.

En cualquier dominio, la propagación del conocimiento[1] por medio de programas de IA se efectúa siempre siguiendo un modelo de razonamiento bien definido. Estos modelos de razonamiento forman parte del motor de inferencias, si hablamos de sistemas de producción, o de las estructuras de control del conocimiento, si hablamos de cualquier otro tipo de sistemas de IA, y contribuyen de manera decisiva a organizar correctamente la búsqueda de soluciones.

Normalmente, las características del dominio y las características de los problemas que deben resolverse condicionan el tipo de modelo de razonamiento que debemos emplear. Así:

- Hay dominios de naturaleza simbólica en los que las soluciones pueden establecerse con "total seguridad"; en estos casos se emplean *modelos categóricos* de razonamiento.
- Hay dominios de naturaleza estadística en los que las soluciones no pueden ser unívocamente obtenidas y en los que es necesario averiguar cuál de las posibles soluciones encontradas es la más probable; en estos casos es preferible razonar con *modelos de naturaleza estadística*, de los cuales el *esquema bayesiano* es el más utilizado.
- Hay otros dominios en los que aparece el concepto de *incertidumbre*, que puede ser inherente a los datos del problema y a los hechos del dominio o a los propios mecanismos inferenciales. En estos casos elegiremos *modelos de razonamiento cuasi-estadísticos*, capaces de manipular correctamente dicha incertidumbre.

[1] Establecimiento de circuitos inferenciales apropiados

- Por último, hay dominios en los que los elementos inferenciales incluyen matices de carácter lingüístico, entre los que pueden establecerse jerarquías y clasificaciones. En estos casos es conveniente emplear *modelos de razonamiento basados en conjuntos difusos*.

$$\begin{aligned} Dominios\_reales &= c_1(categoricos) + c_2(estadisticos) + \\ &\quad c_3(cuasiestadisticos) + c_4(difusos) + \ldots \end{aligned}$$

Esta clasificación no es exhaustiva ni rigurosa (hay más dominios, hay dominios que participan de varias de las características mencionadas,...).

7.1. Modelo Categórico

A continuación trataremos el primero de los modelos comentados en la clasificación inicial del tema: el **modelo categórico**.

7.1.1. Interpretación Diferencial

Una de las grandes cuestiones en la resolución de problemas de IA es cómo utilizar los datos y las verdades demostradas, según un procedimiento encadenado y lógico, para discriminar entre las posibles "soluciones" inicialmente candidatas hasta encontrar la verdadera respuesta del problema planteada.

Cuando el dominio es de naturaleza simbólica, ya se ha comentado que el proceso de razonamiento adecuado debe seguir una aproximación categórica. Uno de tales procedimientos es el de la **interpretación inferencial**, cuyo proceso global sigue aproximadamente un esquema como el siguiente[2]:

1. Recopilación de información *relevante*.
2. Análisis de la importancia relativa de las manifestaciones del problema (ponderación de la información).
3. Análisis de las posibles causas del problema tras considerar, conjunta y razonablemente, todas las manifestaciones del problema. Ello implica el establecimiento tentativo de relaciones causa-efecto (relación de la información disponible con un conjunto de interpretaciones inicialmente posibles).
4. Exclusión una a una de todas aquellas interpretaciones (hipótesis) que no pueden ser explicadas completa y razonablemente por los datos.

[2]Este proceso de razonamiento -sistemático pero complejo- puede simplificarse en función del grado de experiencia, que ayuda en la optimización de recursos al restringir al máximo el conjunto inicial de hipótesis merced al conocimiento heurístico y al efectuar el proceso de establecimiento de relaciones causa-efecto de manera eficaz y eficiente. Por ello, se buscan modelos sistemáticos que utilicen además algún sucedáneo de "sentido común" e "intuición".

5. Fin del proceso con alguno de los siguientes resultados:
   a) Existe una única solución (que ha sido encontrada).
   b) No hay ninguna solución (el conjunto de hipótesis formulado inicialmente no es consistente con los datos).
   c) Hay varias soluciones posibles (hipótesis que se corresponden con los datos) entre las que no se puede discriminar.

7.1.2. Elementos del Razonamiento Categórico

Puesto que una de las tareas que hay que realizar en un proceso de razonamiento categórico es establecer un conjunto de relaciones causales, comenzaremos describiendo nuestro dominio de discurso a partir de dos entidades diferentes, entre las que debemos ser capaces de establecer relaciones:

- **manifestaciones** posibles en el dominio de discurso
- **interpretaciones** posibles en el dominio de discurso

Cualquier dominio no estará completamente descrito, pues, en cuanto no se especifiquen todas las posibles manifestaciones de los problemas que puedan darse en él, todos los posibles problemas del dominio y todas las relaciones causales que puedan establecerse entre problemas y manifestaciones en dicho dominio.

Formalmente, para construir el modelo necesitamos definir una serie de funciones de carácter booleano (ya que el modelo es categórico, en él algo está presente o ausente, es posible o es imposible) que nos sirvan para describir el dominio:

√ Si $x_1, x_2, \ldots, x_n$ es el conjunto completo de todas las manifestaciones posibles del universo de discurso, entonces la función $f(x_1, x_2, \ldots, x_n)$, booleana, asignará el valor 1 a la manifestación x_i si ésta está presente en el problema concreto que nos ocupe, ó 0 en caso contrario (no todas las manifestaciones posibles están presentes en un mismo momento).

√ Si $y_1, y_2, \ldots, y_m$ es el conjunto completo de todas las posibles interpretaciones que se pueden dar a los problemas del dominio, para un problema concreto la función $g(y_1, y_2, \ldots, y_m)$ asignará valor 1 a la interpretación y_j si ésta es posible, ó 0 en caso contrario.

Es necesaria una tercera función, $E = E(X, Y) = E(x_1 x_2 \ldots x_n, y_1 y_2 \ldots y_m)$, **función de conocimiento**, que representa el conjunto de todas las posibles relaciones causales que se pueden establecer en nuestro dominio de discurso, entre manifestaciones e interpretaciones.

Con estos 3 elementos, un problema se reduce a encontrar en un dominio, ante un conjunto f de manifestaciones relacionadas con un problema, la función g que satisface:

$$E : (f \rightarrow g)$$

esto es, encontrar el conjunto de interpretaciones que es compatible con las observaciones y datos de que se dispone, tras la aplicación del conocimiento que se tiene sobre el dominio de discurso.

Este procedimiento, pese a ser eficaz, no es, no obstante, eficiente. Los procedimientos lógicos clásicos, la complejidad de los procesos de resolución, etc. podrían hacer inviable la resolución de determinados problemas. Aparece, por tanto, la necesidad de encontrar alternativas mejores, conceptualmente correctas y computacionalmente eficaces. Una de ellas se describe a continuación.

7.1.3. Procedimiento Sistemático para el R. Categórico

El procedimiento sistemático que se propone para razonar categóricamente consta de las siguientes fases:

1. Identificación de los conjuntos completos de manifestaciones, interpretaciones e función de conocimiento:
   a) construcción del conjunto de todas las combinaciones que se puedan establecer entre las manifestaciones del dominio, *conjunto de complejos de manifestaciones*
   b) construcción del conjunto de todas las combinaciones que se puedan establecer entre las interpretaciones del dominio, *conjunto de complejos de interpretaciones*
   c) construcción del conjunto completo de todas las combinaciones posibles entre complejos de manifestaciones y complejos de interpretaciones, *conjunto de complejos manifestación-interpretación*

   es decir, si hemos identificado n manifestaciones y t interpretaciones en el dominio, el número de complejos de manifestaciones será 2^n, el de complejos de interpretaciones será 2^t y el de complejos manifestación-interpretación será $2^{(n+t)}$.

Dado el carácter exhaustivo del procedimiento, los elementos de los tres conjuntos (que son, como hemos definido, completos) son mutuamente excluyentes. El conjunto de complejos manifestación-interpretación representa el total de situaciones *idealmente posibles* en nuestro universo de discurso, aunque es evidente que no todas ellas vana poder darse en la realidad. Es más, muchas de ellas serán claramente absurdas. El papel del conocimiento será restringir el conjunto total de situaciones idealmente posibles a un conjunto de situaciones *realmente posibles* (permitidas por el propio conocimiento disponible)[3].

Realizado este paso, si nuestro conocimiento sobre el dominio es completo y éste está descrito correctamente, la solución a cualquier problema que podamos plantearnos

[3]En otros términos, pasar de la *base lógica expandida*, BLE, lista exhaustiva de complejos manifestación-interpretación posibles, a la *base lógica reducida*, BLR, lista de complejos manifestación-interpretación compatibles con el conocimiento que se tiene sobre el dominio.

estará en la BLR. Habrá que buscar entre los complejos que la forman aquéllos que presenten las manifestaciones realmente presentes en cada caso (problema) concreto.

Además, la aparición e incorporación de nuevas declaraciones al conocimiento es sencilla, sólo supone recalcular la BLR y revisar las manifestaciones presentes en la nueva lista de complejos.

La situación de no hallar ningún complejo en la BLR que contenga un determinado complejo de manifestaciones tiene tres interpretaciones:

- ▷ las manifestaciones no son realmente las que se están manejando
- ▷ el conocimiento no es correcto, la función E tiene algún error
- ▷ el dominio no está bien construido

El primer problema se resuelve efectuando una nueva recogida de datos para comprobar que no se han cometido fallos al construir la función f. Si el conjunto de manifestaciones sigue siendo el mismo, entonces debe sospecharse de la función de conocimiento (puede ser incompleta, demasiado restrictiva o simplemente falsa) o de la construcción del dominio (que puede contener manifestaciones y/o posibles interpretaciones que no se hayan tenido en cuenta).

El caso contrapuesto, en el que en la BLR aparecen dos complejos manifestación-interpretación que contienen la(s) manifestación(es) concretas del problema en cuestión, permite "acotar" la solución mediante el OR de los mencionados complejos compatibles, pero no resolver el problema por completo. Como regla general, esta situación no es aceptable, y constituye uno de los problemas más serios del modelo que acabamos de desarrollar: *la incertidumbre surge espontáneamente en el modelo categórico.*

Otro problema importante de este modelo es la casi siempre inevitable explosión combinatoria en la construcción de la BLE. Estas y otras deficiencias más sutiles aconsejan la puesta a punto de un modelo alternativo.

7.2. La Corrección Bayesiana

Las interpretaciones categóricas son más bien infrecuentes en el mundo real. Ya hemos comentado que no todos los problemas se manifiestan, algunos no lo hacen nunca y otros tardan mucho en hacerlo. Además, la presencia de una manifestación no siempre es indicativa de algún problema.

Con estas consideraciones, vamos a replantearnos la cuestión desde otro punto de vista: dado un universo y un conjunto de atributos, ¿cuál es la probabilidad de que un determinado elemento del universo presente ciertos atributos del conjunto total?

En términos estadísticos, si N es una determinada población (número de situaciones) y $x_1, x_2, \ldots, x_n$ es el conjunto de todos los atributos posibles, la función $f(x_1, x_2, \ldots, x_n)$ booleana genera un subconjunto de atributos. De este modo, si $N(f)$ es el subconjunto

de los elementos de N que presentan tales atributos, la probabilidad total de f será

$$\textbf{Probabilidad total} = P(f) = \frac{N(f)}{N}$$

El concepto de *probabilidad total* de la estadística bayesiana no es, sin embargo, suficiente para construir un modelo de razonamiento. Necesitamos introducir el concepto de *probabilidad condicional* (la probabilidad de las causas).

En la probabilidad condicional aparecen involucrados dos sucesos, de forma qe la ocurrencia del segundo depende de la ocurrencia del primero. Este planteamiento se acerca algo más a los aspectos relativos al *razonamiento*. Sin embargo, lo que interesa en principio es interpretar cosas que ya han pasado, para lo cual necesitamos introducir, además del mecanismo "a priori", algún mecanismo para razonar "a posteriori". Es decir, no sólo saber con qué probabilidad se dará una consecuencia conocidas las probabilidades de sus causas, sino, conocida la probabilidad de una consecuencia, saber la probabilidad de sus causas.

Este fue, precisamente, el planteamiento del reverendo Bayes, que se tradujo en la ecuación elemental de su famoso *teorema*:

$$p(A/E) = \frac{p(E/A)p(A)}{p(E)}$$

que permite obtener la probabilidad condicional "a posteriori" a partir de la probabilidad condicional "a priori" y de las probabilidades totales. Esta expresión debe poder generalizarse para el análisis de problemas más complicados. Supóngase la siguiente situación:

	A	¬A	Total
E	a	b	$a+b$
¬E	c	d	$c+d$
Total	$a+c$	$b+d$	N

donde

$$\begin{aligned} a &= \text{verdaderos positivos} \\ b &= \text{falsos positivos} \\ c &= \text{falsos negativos} \\ d &= \text{verdaderos negativos} \end{aligned}$$

A partir de estos datos pueden establecerse las siguientes probabilidades totales o *prevalencias*:

$$p(E) = \frac{(a+b)}{N} \qquad p(\neg E) = \frac{(c+d)}{N}$$

$$p(A) = \frac{(a+c)}{N} \qquad p(\neg A) = \frac{(b+d)}{N}$$

y las siguientes probabilidades condicionales:

$$(1) \quad p(E/A) = \frac{a}{a+c} \qquad (5) \quad p(A/E) = \frac{a}{a+b}$$

$$(2) \quad p(E/\neg A) = \frac{b}{b+d} \qquad (6) \quad p(A/\neg E) = \frac{c}{c+d}$$

$$(3) \quad p(\neg E/A) = \frac{c}{a+c} \qquad (7) \quad p(\neg A/E) = \frac{b}{a+b}$$

$$(4) \quad p(\neg E/\neg A) = \frac{d}{b+d} \qquad (8) \quad p(\neg A/\neg E) = \frac{d}{c+d}$$

(1) suele denominarse *sensibilidad* y (4) *especifidad*; suele interesar conseguir la máxima sensibilidad y a la vez la máxima especifidad.

Por el teorema de Bayes sabemos que

$$p(A/E) = \frac{p(E/A)p(A)}{p(E)}$$

siendo el término denominador el que "impide" la generalización; también sabemos que

$$p(E) = \frac{(a+b)}{N}$$

de la expresión (1) se deduce

$$a = (a+c)p(E/A)$$

y de la expresión (2)

$$b = (b+d)p(E/\neg A)$$

Por otra parte,

$$p(A) = \frac{(a+c)}{N}$$

y

$$p(\neg A) = \frac{(b+d)}{N}$$

por lo que

$$(a+c) = N \cdot p(A)$$

y

$$(b+d) = N \cdot p(\neg A)$$

y sustituyendo

$$a = N \cdot p(A) \cdot p(E/A)$$

$$b = N \cdot p(\neg A) \cdot p(E/\neg A)$$

Si sustituimos a su vez en la expresión de $p(E)$ resulta

$$p(E) = p(A)p(E/A) + p(\neg A)p(E/\neg A)$$

Y llevando este resultado a la expresión original del teorema de Bayes, obtenemos

$$p(A/E) = \frac{p(E/A)p(A)}{p(E/A)p(A) + p(E/\neg A)(\neg A)}$$

ecuación que es directamente generalizable, dando lugar a la *expresión generalizada del teorema de Bayes*:

$$p(A_0/E) = \frac{p(E/A_0)p(A_0)}{\sum_i p(E/A_i)p(A_i)}$$

si consideramos más de dos posibilidades (en lugar de simplemente A y $\neg A$).

Así pues, si el tratamiento lógico de un problema concreto es correcto, si los conjuntos de manifestaciones e interpretaciones son completos, si las manifestaciones y las interpretaciones cumplen el requisito de independencia exigido por el teorema de Bayes, si la función de conocimiento está bien construida y si el tratamiento estadístico efectuado es correcto, entonces sólo las probabilidades condicionales relativas a complejos manifestación-interpretación que aparecen en la BLR tendrán valores distintos de cero. De no ser así, deberá pensarse en alguna de las siguientes deficiencias:

- ▷ el planteamiento lógico no es correcto
- ▷ la función de conocimiento no es correcta (es incompleta o está mal construida)
- ▷ la estadística no ha sido bien realizada

Además, para asegurar la consistencia matemática del modelo se tiene que cumplir que

$$\sum_i P(m_i/i_0) = 1$$

De esta manera, la solución más *probable* de entre los complejos manifestación-interpretación presentes en la BLR que aporten la manifestación real será sencillamente la que tenga mayor probabilidad.

A pesar de todo, la corrección bayesiana también presenta problemas: si manifestaciones e interpretaciones no son independientes, el modelo bayesiano fracasa[4]. Esta limitación del modelo plantea problemas cuando se pretende su aplicación en dominios del mundo real, en los que los requisitos de independencia casi nunca se cumplen.

[4]Ojo: las manifestaciones han de ser independientes entre sí y las interpretaciones independientes entre sí, pero no han de ser manifestaciones independientes de interpretaciones, o de lo contrario ¡no habría relaciones causales!

Pero ésta no es la única deficiencia del modelo bayesiano. En los problemas interesantes para la aplicación de técnicas de IA, la información suele ir apareciendo progresivamente, secuencialmente y, generalmente, de forma poco ordenada. En estos casos, adecuar la aproximación bayesiana a la interpretación secuencial supone considerar que la información factual aparece incrementalmente y, por lo tanto, habrá que adaptar las ecuaciones correspondientes.

Sea E_1 el conjunto de toda la información disponible en un momento dado, y sea S_1 un nuevo dato (un nuevo elemento de información que acaba de aparecer). Entonces E será el nuevo conjunto formado por la información de E_1 y el nuevo dato S_1. La ecuación del teorema de Bayes se reescribe:

$$p(I_i/E) = \frac{p(S_1/I_i \text{ y } E_1)p(I_i/E_1)}{\sum_j p(S_1/I_j \text{ y } E_1)p(I_j/E_1)}$$

reescritura que complica aún más la estadística asociada a la aplicación del modelo bayesiano.

Otro problema frecuente de este modelo procede de su aplicación poco cuidadosa, y un último gran problema es consecuencia de su consistencia matemática. Al respecto, y por definición, siempre se tiene que cumplir que

$$p(A) + p(\neg A) = 1$$

Sin embargo, cuando tratamos con problemas del mundo real, difícilmente puede asumirse esta consistencia, es decir, que un mismo conjunto de evidencias apoye simultáneamente (aunque en distinto grado) una hipótesis y su negación. Este es uno de los puntos más débiles de los modelos estadísticos cuando tratamos con "conocimiento" en lugar de tratar con "datos". En problemas del mundo real, dada una proposicón basada en la experiencia, la consistencia matemática no tiene por qué mantenerse. Necesitamos la puesta a punto de un nuevo modelo, y al efecto surgirán los *modelos cuasiestadísticos*[5] como **factores de certidumbre** y la **teoría evidencial** de Dempster y Shafer.

[5]Usan con profusión heurísticas para paliar los defectos de los modelos estadísticos.

Capítulo 8

Factores de Certidumbre

Parte de los inconvenientes encontrados en los modelos vistos en capítulos anteriores podrían ser resueltos empleando conocimiento heurístico. Si más concretamente nos referimos a los problemas estadísticos derivados de la inevitable y exhaustiva recolección de datos, una posible solución podría pasar por el establecimiento de un nuevo concepto, el de *probabilidad condicional subjetiva*, que definimos como una medida numérica que relaciona dos sucesos, de forma que la ocurrencia de uno está condicionada por la ocurrencia del otro, pero donde la relación no está avalada por amplios estudios estadísticos.

Así, la expresión

$$p(I_i/S_k) = x$$

podría traducirse como "SI *la manifestación* S_k *está presente,* ENTONCES *según mi experiencia hay una probabilidad (subjetiva)* x *de que la interpretación sea* I_i".

Este enfoque resuelve el problema de la recogida masiva de información y datos, pero sigue presentando problemas, ya que es posible que la suma de las probabilidades condicionales subjetivas sea distinta de la unidad

$$\sum_i p(I_i/S_k) \neq 1$$

situación que no es compatible con la estadística tradicional, pero que puede resolverse *normalizando* a uno las correspondientes probabilidades condicionales subjetivas, con el único objetivo de mantener la consistencia matemática del modelo.

Otros problemas, sin embargo, no son tan fáciles de resolver, ya que cuando trabajamos con información de carácter simbólico en lugar de con datos numéricos aparecen conceptos como *imprecisión*, *incertidumbre*, *falta de información*, *credibilidad*,... que son difíciles de definir. La necesidad de construir programas inteligentes capaces de manipular estos tipos diferentes de información sugiere la conveniencia de formalizar nuevas aproximaciones y modelos.

8.1. Modelo de los Factores de Certidumbre

La expresión $p(I_i/S_k) = x$ puede interpretarse en términos de implicación como

$$S_k \xrightarrow{x} I_i$$

expresión en la que x define la llamada **potencia evidencial** de la implicación. Cuando $x = 1$ se dice que la relación entre S_k e I_i es *patognómica* (de absoluta certeza, sin ninguna incertidumbre); en otro caso, para $x \geq 0$ y $x < 1$ la *potencia evidencial* establece la *intensidad* de la relación causal.

A lo largo de un proceso completo de razonamiento, y manteniendo I_i como hipótesis de trabajo, seguramente aparecerá evidencia a favor de la hipótesis considerada, pero también en contra de la misma. Si aplicamos un esquema bayesiano al problema planteado, deberíamos concluir también que

$$S_k \xrightarrow{1-x} \neg I_i$$

lo cual, como hemos visto, cuando trabajamos con conocimiento es inaceptable.

En 1975, para tratar de resolver este tipo de problemas, Shortliffe y Buchanan plantearon un modelo de razonamiento de naturaleza "ad hoc" que sacudió los cimientos del entonces incipiente mundo de la IA por su consiguiente carencia de una base teórica fuerte. No obstante, fue inmediatamente aceptado debido a su fácil comprensión y a la calidad de los resultados obtenidos tras su aplicación. Sus ideas básicas pueden resumirse en:

- Dada una hipótesis que está siendo considerada, la potencia evidencial de una declaración debe representarse a través de dos medidas diferentes: la **medida de confianza creciente**, MB y la **medida de desconfianza creciente**, MD, que son en realidad índices dinámicos que representan incrementos asociados a evidencias nuevas.
- Si h es una hipótesis y e una evidencia, la misma evidencia e no puede, simultáneamente, incrementar y disminuir la confianza en h.
- $MB(h, e)$ representa el incremento de la confianza en h dada la evidencia e.
- $MD(h, e)$ representa el incremento de la desconfianza en h dada la evidencia e.

Con estas premisas podemos establecer el siguiente formalismo y casos particulares: sea $p(h)$ la confianza previa en h antes de aparecer e, $p(h/e)$ la confianza en h tras la aparición de e y $1 - p(h)$ la desconfianza previa en h antes de aparecer e,

1. Si $p(h/e) > p(h)$, entonces la nueva evidencia produce un aumento de confianza en la hipótesis considerada. En este caso,
   - $MB(h, e) > 0$
   - $MD(h, e) = 0$

   y $MB(h, e)$ se define como el incremento de la confianza entre un factor normalizador (si $p(h/e) = 1$, $MB(h, e)$ debe ser 1):

- $MB(h,e) = \dfrac{p(h/e) - p(h)}{1 - p(h)}$

De modo que $MB(h,e)$ representa el incremento relativo de la confianza en la hipótesis h tras la aparición de la evidencia e, que coincide con la disminución relativa de la desconfianza en h tras la aparición de la evidencia e.

2. Si $p(h) > p(h/e)$, entonces la nueva evidencia produce un aumento de la desconfianza[1] depositada en la hipótesis considerada. En este caso,
   - $MB(h,e) = 0$
   - $MD(h,e) > 0$

   y $MD(h,e)$ se define
   - $MD(h,e) = \dfrac{p(h) - p(h/e)}{p(h)}$

   De modo que $MD(h,e)$ representa el incremento relativo de la desconfianza en la hipótesis h tras la aparición de la evidencia e, que coincide con la disminución relativa de la confianza en h tras la aparición de la evidencia e.
3. Si $p(h/e) = p(h)$, entonces la nueva evidencia es independiente de la hipótesis considerada, ya que no aumenta ni la confianza ni la desconfianza. En este caso,

$$MB(h,e) = MD(h,e) = 0$$

Claramente, si $p(h)$ simboliza una probabilidad a priori en sentido clásico, podemos establecer los valores límite de las correspondientes medidas de confianza y desconfianza crecientes según las expresiones:

$$MB(h,e) = \begin{cases} 1 & \text{si} \quad p(h) = 1 \text{ (certeza absoluta)} \\ \dfrac{max[p(h/e), p(h)] - p(h)}{max[1,0] - p(h)} & \quad p(h) \neq 1 \end{cases}$$

$$MD(h,e) = \begin{cases} 1 & \text{si} \quad p(h) = 0 \text{ (falsedad absoluta)} \\ \dfrac{min[p(h/e), p(h)] - p(h)}{min[1,0] - p(h)} & \quad p(h) \neq 0 \end{cases}$$

Ambas expresiones no son más que representaciones formales y simétricas de las medidas de confianza y desconfianza crecientes, expresadas en términos de probabilidades condicionales y de probabilidades a priori.

Además de estas dos medidas, Shortliffe y Buchanan definen un tercer índice, denominado **factor de certidumbre**, CF, que combina las dos anteriores:

[1] Ojo: ¡no disminución de la confianza!

$$CF(h,e) = MB(h,e) - MD(h,e)$$

expresión que también es de carácter formal (carece de entidad propia, coincide con MB ó con MD), ya que una misma evidencia nunca puede incrementar, simultáneamente, la confianza y la desconfianza en la misma hipótesis.

Shortliffe y Buchanan justifican su introducción como un medio para facilitar la comparación entre potencias evidenciales de hipótesis alternativas en relación a una misma evidencia.

En cada una de las tres medidas desarrolladas podemos identificar las siguientes características:

- RANGOS

$$\begin{array}{rcccl} 0 & \leq & MB(h,e) & \leq & 1 \\ 0 & \leq & MD(h,e) & \leq & 1 \\ -1 & \leq & CF(h,e) & \leq & 1 \text{ (“no sé” se dice siempre “0”)} \end{array}$$

- COMPORTAMIENTO EN CASOS EXTREMOS E HIPÓTESIS MUTUAMENTE EXCLUYENTES
  Si h es cierta ($p(h/e) = 1$),

$$\begin{array}{rcl} MB(h,e) & = & \dfrac{1-p(h)}{1-p(h)} = 1 \\ MD(h,e) & = & 0 \\ CF(h,e) & = & 1 \end{array}$$

  Si $\neg h$ es cierta ($p(\neg h/e) = 1$),

$$\begin{array}{rcl} MB(h,e) & = & 0 \\ MD(h,e) & = & \dfrac{0-p(h)}{0-p(h)} = 1 \\ CF(h,e) & = & -1 \end{array}$$

  es decir, $MB(\neg h, e) = 1$ si y sólo si $MD(h,e) = 1$, resultado que se obtiene sin más que recordar cómo se definen MB y MD. Por otra parte, si h_1 y h_2 son hipótesis mutuamente excluyentes y $MB(h_1, e) = 1$, entonces podemos afirmar rotundamente que $MD(h_2, e) = 1$ (algo que será útil para propagar conocimiento).

- EVIDENCIAS INDEPENDIENTES DE LA HIPÓTESIS
  Sea h la hipótesis considerada y e una evidencia; si e es independiente de h (ni la apoya ni va en su contra), entonces

$$\begin{array}{rcl} p(h/e) & = & p(h) \\ MB(h,e) & = & 0 \\ MD(h,e) & = & 0 \\ CF(h,e) & = & 0 \end{array}$$

- DIFERENCIA ENTRE FACTORES DE CERTIDUMBRE Y PROBABILIDADES CONDICIONALES
  Uno de los puntos más débiles de los modelos probabilísticos era el hecho de que una misma evidencia apoyaba simultáneamente una hipótesis y su negación, como consecuencia de la consistencia matemática de tales modelos ($p(h/e) + p(\neg h/e) = 1$).

  Este inconveniente no aparece en este modelo, cuyos autores afirman textualmente "... los factores de certidumbre de las hipótesis h y $\neg h$ no son complementarios a la unidad, son opuestos entre sí"; así, si el apoyo que una evidencia presta a una hipótesis es bajo, no debe ser alto el apoyo a su negación, sobre todo si la información no es completa (el apoyo a ambas, en ese caso debe ser bajo). Tal afirmación se demuestra analizando casos extremos:

  Si $p(h/e) > p(h)$, entonces $[p(\neg h/e) = 1 - p(h/e)] < [1 - p(h) = p(\neg h)]$, de modo que

$$\begin{array}{rcl} MB(\neg h, e) & = & 0 \\ MD(\neg h, e) & > & 0 \\ MB(h, e) & > & 0 \\ MD(h, e) & = & 0 \end{array}$$

  pero

$$\begin{aligned} CF(\neg h, e) &= MB(\neg h, e) - MD(\neg h, e) = \\ &= \frac{0 - [p(\neg h) - p(\neg h/e)]}{p(\neg h)} = \frac{[p(\neg h) - p(\neg h/e)]}{p(\neg h)} = \\ &= \frac{1 - p(h/e) - 1 + p(h)}{1 - p(h)} = \frac{p(h) - p(h/e)}{1 - p(h)} = \\ &= -MB(h, e) = -CF(h, e) \end{aligned}$$

  por lo que $CF(\neg h, e) = -CF(h, e)$. El mismo resultado se obtiene si consideramos el otro caso extremo, en el que $p(h) > p(h/e)$.

8.2. Combinación de Evidencias

¿Cómo manejar los factores de certidumbre? En el caso concreto de una única evidencia la respuesta es clara:

- ↪ Se indicará un valor mayor que cero y menor o igual a uno si la evidencia en cuestión apoya la hipótesis.
- ↪ Se indicará un valor menor que cero, pero mayor o igual a menos uno, si la evidencia en cuestión va en contra de la hipótesis.

↪ Se indicará un valor igual a cero si se estima que la evidencia encontrada no tiene nada que ver con la hipótesis considerada.

El problema, no obstante, se complica cuando hay más de una evidencia relativa a una misma hipótesis. En ese caso hablamos de *combinación de evidencias* que afectan a una misma hipótesis. El problema planteado puede formularse en los siguientes términos: "Sea un conjunto de reglas, todas ellas con la misma conclusión, cada una de las cuales viene afectada de un factor de certidumbre diferente, ¿cuál es el factor de certidumbre resultante, considerando toda la evidencia?"

$$\begin{array}{c}\text{IF } e_1 \text{ THEN } H \text{ WITH } CF(H, e_1)\\ \text{IF } e_2 \text{ THEN } H \text{ WITH } CF(H, e_2)\\ \ldots\\ \text{IF } e_n \text{ THEN } H \text{ WITH } CF(H, e_n)\end{array}$$

Los factores de certidumbre de las distintas reglas pueden interpretarse como las potencias evidenciales de las relaciones causales correspondientes:

$$\begin{array}{c} e_1 \overset{CF(H,e_1)}{\longrightarrow} H\\ e_2 \overset{CF(H,e_2)}{\longrightarrow} H\\ \ldots\\ e_n \overset{CF(H,e_n)}{\longrightarrow} H\end{array}$$

Cada una de las evidencias contribuye, favorable o desfavorablemente, al establecimiento de la veracidad de la hipótesis considerada. El problema estriba en encontrar una formulación adecuada que permita evaluar $CF(H, E)$ donde $E = e_1 \wedge e_2 \wedge \ldots \wedge e_n$ (toda la evidencia).

Shortliffe y Buchanan proponen una primera aproximación para la combinación entre pares de evidencias que se refieren a la misma hipótesis, considerando los tres casos posibles:

1. Si e_1 y e_2 contribuyen positivamente a la veracidad de la hipótesis H, entonces
   - ⋆ $CF(H, e_1) > 0$
   - ⋆ $CF(H, e_2) > 0$
   - ⋆ $CF(H, e_1 \wedge e_2) = CF(H, e_1) + CF(H, e_2) - [CF(H, e_1) \times CF(H, e_2)]$

   (se resta la "intersección" para no tenerla en cuenta dos veces)
2. Si e_1 y e_2 contribuyen negativamente a la veracidad de la hipótesis H, entonces
   - ⋆ $CF(H, e_1) < 0$
   - ⋆ $CF(H, e_2) < 0$
   - ⋆ $CF(H, e_1 \wedge e_2) = CF(H, e_1) + CF(H, e_2) + [CF(H, e_1) \times CF(H, e_2)]$

3. Si e_1 contribuye positivamente a la veracidad de la hipótesis H y e_2 contribuye negativamente, entonces
   - $\star$ $CF(H, e_1) > 0$
   - $\star$ $CF(H, e_2) < 0$
   - $\star$ $CF(H, e_1) \times CF(H, e_2) < 0$
   - $\star$ $CF(H, e_1 \wedge e_2) = CF(H, e_1) + CF(H, e_2)$

   (se impone el que más influya)

Esta primera aproximación es coherente con la idea de que, ante la posibilidad de información incompleta, el efecto conjunto de dos evidencias debe ser igual a la suma de sus efectos por separado menos su efecto conjunto (en el caso de contribución positiva). En otras palabras, nos previene de la hipotética situación de que ambas evidencias pudieran no ser completamente independientes (caso en el que la mencionada "intersección" sería no vacía).

Además, las expresiones anteriores son directamente generalizables, pues si en lugar de dos evidencias tenemos n, todas ellas (por ejemplo) con CFs mayores que cero, el efecto conjunto de todas ellas sobre la hipótesis H responde a la expresión

$$CF(H, E) = \sum_{i}^{n} CF_i - \sum_{\substack{i \\ i<j}}^{n} CF_i CF_j + \sum_{\substack{i \\ i<j<k}}^{n} CF_i CF_j CF_k - \ldots$$

(obsérvese la alternancia de signos, que asegura que eliminamos las componentes de intersección, y que las sucesivas condiciones de las sumas evitan duplicar los productos de los factores de certidumbre involucrados).

Esta aproximación parece razonable y no tiene ninguna fisura teórica. No obstante, fueron los propios Shortliffe y Buchanan quienes inmediatamente propusieron un modelo alternativo debido a la falta de asociatividad de la formulación y sus consecuencias.

En primer lugar, el orden en que aparecen las evidencias modifica considerablemente el resultado final, aunque ello puede ser una ventaja en dominios en los que hay relaciones temporales y para los que el orden de aparición de las evidencias es realmente importante. La segunda objeción tiene que ver con la gran sensibilidad de la formulación ante la aparición de evidencias contradictorias en estados avanzados del proceso de razonamiento.

Ni Shortliffe ni Buchanan consideraron aceptable esta situación y propusieron una segunda aproximación que paliaba estas "deficiencias":

$*$ Si $CF(H, e_1) > 0$ y $CF(H, e_2) > 0$, entonces

$$CF(H, e_1 \wedge e_2) = CF(H, e_1) + CF(H, e_2) - [CF(H, e_1) \times CF(H, e_2)]$$

$*$ Si $CF(H, e_1) < 0$ y $CF(H, e_2) < 0$, entonces

$$CF(H, e_1 \wedge e_2) = CF(H, e_1) + CF(H, e_2) + [CF(H, e_1) \times CF(H, e_2)]$$

* Si $CF(H, e_1) \times CF(H, e_2) < 0$, entonces

$$CF(H, e_1 \wedge e_2) = \frac{CF(H, e_1) + CF(H, e_2)}{1 - min\{|CF(H, e_1)|, |CF(H, e_2|\}}$$

Esta nueva forma de combinar evidencias referidas a una misma hipótesis sí que es asociativa y, por lo tanto, las evidencias se pueden considerar en cualquier orden sin que el resultado final se vea afectado[2]. Además presenta la ventaja de permitir modelizar procesos de razonamiento sin tener que almacenar explícitamente los MBs y los MDs.

Quedan, no obstante, algunos problemas por resolver. Por ejemplo, el modelo supone implícitamente independencia condicional de las evidencias. Por ello, si e_1 implica lógicamente a e_2, entonces $CF(H, e_1 \wedge e_2)$ debería ser igual a $CF(H, e_1)$, pero de la aplicación del modelo no se deduce este resultado, lo que constituye un problema no resuelto de la combinación de evidencias. Al respecto, Shortliffe y Buchanan proponen algunas alternativas como estructurar muy bien las bases de conocimientos o agrupar en una sola regla cláusulas con evidencias condicionalmente dependientes. En todo caso, tales soluciones pertenecen más al ámbito de la ingeniería del conocimiento que al de la IA.

8.3. Propagación de Incertidumbre

Hasta ahora hemos considerado que la evidencia relacionada con una hipótesis era un hecho (a favor o en contra de la hipótesis) que no venía afectado de incertidumbre. Por ello, el factor de certidumbre correspondiente $CF(h, e)$ podía ser interpretado como la potencia evidencial de la implicación $e \rightarrow h$ de forma que

- $\surd$ si $CF(h, e) = 1$, la evidencia e implica lógicamente a la hipótesis h
- $\surd$ si $CF(h, e) = -1$, la evidencia e implica lógicamente a la negación de la hipótesis h
- $\surd$ si $CF(h, e) = 0$, la evidencia e es independiente de la hipótesis h

Sin embargo, cuando tratamos de representar conocimiento para luego llevar a cabo procesos de razonamiento, lo más normal y lo más correcto es tratar de establecer relaciones causales basadas en "hechos inciertos". Es decir, si establecemos la relación causal imprecisa

$$\text{SI } e \text{ ENTONCES } h \text{ con } CF(h, e)$$

la imprecisión asociada está referida a la implicación subyacente (siempre que tengamos e podremos concluir h con una incertidumbre asociada dada por el facor de certidumbre); lo normal, sin embargo, es que en los problemas reales la propia evidencia venga afectada de una cierta imprecisión, que en el curso de un proceso inferencial contribuirá a modificar la incertidumbre de las conclusiones alcanzadas.

[2]Es el denominador el que convierte el modelo en asociativo prescindiendo de la relación de orden.

Nótese que hay una sutil diferencia entre **imprecisión** e **incertidumbre**. La *imprecisión* es una característica que afecta a las entidades, hechos y/o datos del dominio, mientras que la *incertidumbre* es una característica ligada a los procesos de razonamiento.

En cualquier caso, si basamos nuestro proceso de razonamiento en hechos imprecisos, las conclusiones que obtengamos serán inciertas, no podrán establecerse con total certidumbre. Y esto justamente es uno de los problemas más cruciales de la IA: la **propagación de la incertidumbre**[3].

Este problema surge, fundamentalmente, por dos circunstancias que pueden darse aislada o conjuntamente:

- ▷ La propia evidencia es imprecisa
- ▷ La evidencia considerada es consecuencia de otra regla y forma parte de un proceso de razonamiento que supone varias inferencias (arrastra la imprecisión)

Shortliffe y Buchanan proponen un esquema en el que la primera circunstancia puede considerarse como un caso particular de la segunda.

De este modo, si tenemos

$$\text{SI } E' \text{ ENTONCES } E \text{ con } CF(E, E') = x$$
$$\text{y}$$
$$\text{SI } E \text{ ENTONCES } H \text{ con } CF(H, E) = y$$

se genera el circuito inferencial

$$E' \xrightarrow{x} E \xrightarrow{y} H$$

y para resolverlo postulan

$$CF(H, E') = CF(H, E) \times max[0, CF(E, E')]$$

Esta formulación presenta una dificultad no prevista: cuando $CF(E, E')$ es menor que cero y por tanto E' apoya la negación de E, la ecuación proporciona un $CF(H, E') = 0$ que impide afirmar nada (ni positivo ni negativo) sobre la hipótesis considerada[4].

Este inconveniente del modelo fue estudiado por Heckerman, discípulo de Shortliffe y Buchanan, quien propuso una formulación alternativa basándose en el siguiente razonamiento: Dado que $CF(\neg E, E') = -CF(E, E')$, la línea de razonamiento de nuestro circuito inferencial debería poder modificarse en los siguientes términos

$$E' \longrightarrow \neg E \longrightarrow H$$

[3]Que está ligado al concepto de *entropía de la información*, según el cual la información tiende a degradarse en la medida que es utilizada.

[4]Esto es, si hay evidencias contradictorias se contesta siempre "no sé".

en lugar de

$$E' \longrightarrow E \longrightarrow H$$

para aquellos casos en los que $CF(E', E) < 0$. Así, directamente, podemos obtener el factor de certidumbre buscado,

$$CF(H, E') = CF(H, \neg E) \times max[0, CF(\neg E, E')]$$

La modificación de Heckerman puede resumirse del siguiente modo:

$$CF(H, E') = CF(h, E) \times CF(E, E') \quad \Leftrightarrow \quad CF(E, E') \geq 0$$

$$CF(H, E') = CF(h, \neg E) \times CF(\neg E, E') \quad \Leftrightarrow \quad CF(E, E') < 0$$

El esquema supuestamente alternativo de Heckerman no es, no obstante, más que una aplicación estricta del modelo de Shortliffe y Buchanan, que adolece por otra parte del mismo problema que hemos comentado en relación al esquema bayesiano: precisamente Shortliffe y Buchanan establecieron que $CF(H, E) = -CF(\neg H, E)$ como un argumento en contra de la consistencia matemática de los modelos estadísticos cuando, en lugar de datos, empleamos conocimiento[5].

Un último aspecto relacionado con la propagación de incertidumbre es el relativo a la **combinación lógica de evidencias**. Recordemos que en los sistemas de producción las cláusulas de los antecedentes de las reglas solían estar anidadas a través de operadores lógicos AND, OR y NOT. ¿Cómo se obtienen los valores de expresiones como $CF(H_1 \text{ AND } H_2, E)$ o $CF(H_1 \text{ OR } H_2, E)$ a partir de $CF(H_1, E)$ y $CF(H_2, E)$ donde E es toda la evidencia del antecedente?

Evidentemente, el procedimiento seguido debe pasar por evaluar primero el antecedente e inferir luego la conclusión teniendo en cuenta la potencia evidencial de la declaración correspondiente. Así, si E representa la evidencia disponible (que puede no coincidir exactamente con la de la declaración) y teniendo en cuenta la propia regla,

$$\begin{aligned} CF(H, E) \;=\; & CF(H, [E_1 \text{ AND } (E_2 \text{ OR } E_3) \text{ AND } E_4]) \\ & \times \\ & CF([E_1 \text{ AND } (E_2 \text{ OR } E_3) \text{ AND } E_4], E) \end{aligned}$$

expresión que tiene en cuenta simultáneamente la potencia evidencial de la declaración (primer factor) y la imprecisión en la información disponible de acuerdo con la estructura del antecedente de la regla (segundo factor).

Las funciones propuestas para evaluar los efectos de las conjunciones y de las disyunciones son:

$$CF(H_1 \text{ AND } H_2, E) = min\{CF(H_1, E), CF(H_2, E)\}$$

$$CF(H_1 \text{ OR } H_2, E) = max\{CF(H_1, E), CF(H_2, E)\}$$

[5]No obstante, no Shortliffe ni Buchanan dan demasiada importancia a esta peculiaridad de su modelo y siguen pensando que de la afirmación de un hecho no debe derivarse también una cuantificación para la negación del hecho afirmado.

Capítulo 9

Teoría Evidencial

A diferencia del modelo de los factores de certidumbre de Shortliffe y Buchanan, el esquema de razonamiento propuesto por Dempster y Shafer sí tiene una fuerte base teórica[1]. Nosotros, no obstante, veremos una simplificación notable, aunque no por ello menos rigurosa, de la misma.

El esquema de razonamiento que proponen Dempster y Shafer en su **teoría evidencial** es atractivo, entre otra razones:

- porque permite modelizar de forma sencilla la incertidumbre asociada a evidencias e hipótesis
- porque permite considerar conjuntos de hipótesis sin que la confianza depositada en cada uno de ellos tenga que ser distribuida de ningún modo entre cada una de las hipótesis individuales del conjunto[2]
- porque permite reflejar de forma elegante la falta de conocimiento tan frecuentemente ligada a los procesos de razonamiento
- porque contiene a la teoría de la probabilidad como un caso particular
- porque contiene a algunas de las funciones combinatorias de evidencias del modelo de los factores de certidumbre (concretamente, a aquéllas que funcionan bien en el modelo de Shortliffe y Buchanan)[3]

La cuestión es, ¿cómo maneja este nuevo modelo el conocimiento inexacto y la falta de conocimiento?

[1] La **teoría evidencial** o *teoría de lo desconocido* fue, de hecho, inicialmente, un modelo de razonamiento propuesto por Dempster que se convirtió en toda una teoría tras la formalización de Shafer.

[2] Una de las novedades, como vemos, es que el concepto de *hipótesis* se generaliza a "grupo de hipótesis", de modo que lo que aquí pretende decirse es que el apoyo de una evidencia no discrimina entre ellas.

[3] Es decir, funciones combinatorias donde dos o más evidencias se refieren a la misma hipótesis, no cadenas de evidencias del estilo $E \rightarrow E' \rightarrow H$, donde hemos visto que hay problemas.

9.1. La Teoría Evidencial de Dempster y Shafer

En primer lugar, dado un universo de discurso cualquiera, se introduce el concepto de **marco de discernimiento**, que se define como "el conjunto finito de todas las hipótesis que se pueden establecer en el dominio del problema", esto es, todas las posibles soluciones que se pueden dar en él (de las que se asume que una y sólo una es la correcta). Este *marco de discernimiento* forma un conjunto completo, y por tanto exhaustivo, de hipótesis mutuamente excluyentes.

Por otra parte, como ya ha sido esbozado, el efecto de una determinada evidencia sobre el conjunto global de hipótesis no viene determinado por la contribución de la confianza depositada en las hipótesis individuales. Por el contrario, el efecto de cada evidencia afecta generalmente a un subconjunto de hipótesis del *marco de discernimiento*, planteamiento totalmente coherente con la realidad de casi todos los problemas reales. En éstos, las evidencias prácticamente nunca confirman sólo una hipótesis, sino que lo más normal es que permitan discriminar entre grupos de hipótesis alternativas, manteniéndose la incertidumbre entre las propias hipótesis individuales.

Así pues, según esta teoría, se define

$\theta = \{H_1, H_2, \ldots, H_n\}$	marco de discernimiento
$H_1, H_2, \ldots, H_n$	hipótesis del marco
A	subconjunto cualquiera del marco
$\Gamma(\theta)$	conjunto de todos los subconjuntos posibles del marco (partes de)

En este contexto, la aparición de una determinada evidencia e que favorezca a un subconjunto A de θ ($A \subseteq \theta$) en un determinado grado representado por $m_e(A)$. Dicho m_e, que toma valores en el intervalo cerrado $[0, 1]$ ($m_e \in [0, 1]$), es el indicativo de la confianza que la evidencia e permite depositar en A, se denomina **función básica de asignación de verosimilitud**. Todo esto se representa:

$$e : A = \{h_a, h_b, h_c\} \rightarrow m_e(A) = x, \text{ con } x \in [0, 1]$$

Además, el $1 - x$ restante no se asigna a $\neg A$, sino a θ, ya que no podemos saber realmente dónde va a parar la confianza de esa evidencia, de cuantía $1 - x$ (modelo de incertidumbre).

Visto de otro modo, en ausencia de toda evidencia se tiene que

$$m_e(\theta) = 1$$

ya que θ es, por definición, el conjunto de todas las soluciones posibles tras análisis del dominio. Si entonces surge una evidencia e relativa a un $A \subseteq \theta$ tal que

$$m_e(A) = x$$

entonces es lógico pensar que se reduce

$$m_e(\theta) = 1 - x$$

Incluso algebraicamente, pensando en Γ como conjunto "partes de"[4] θ, si tenemos un $A \subseteq \theta \Rightarrow A \in \Gamma(\theta)$ que es apoyado en x por una evidencia e, no tiene sentido realmente intentar hablar del complementario de A para asignarle la confianza $1 - x$.

Además, se definen las siguientes condiciones para la *función básica de asignación de verosimilitud*:

- $\surd$ la suma extendida a todos los subconjuntos del marco de todas las funciones de asignación de verosimilitud vale 1 (ya que la solución está necesariamente en el marco)

$$\sum_{A \subset \theta} m(A) = 1$$

- $\surd$ la función básica de asignación de verosimilitud del conjunto vacío vale 0 (por la misma razón anterior)

$$m(\varnothing) = 0$$

De manera que, volviendo a lo que comentábamos anteriormente, la forma elegante que la teoría evidencial proporcionaba para tratar la falta de conocimiento asociada a los procesos de razonamiento, puede expresarse:

$$e : A \subset \theta \rightarrow m(A) = x, \text{ con } 0 \leq x \leq 1$$

y ya que

$$\sum_{A \subset \theta} m(A) = 1$$

el resto de la confianza que no ha sido asignada a A corresponde a

$$m(\theta) = 1 - m(A) = 1 - x$$

Todo subconjunto del *marco de discernimiento* para el cual exista una evidencia e tal que verifique $m_e(A) \neq 0$ se denomina **elemento focal**. Así pues, puesto que la evidencia e supone la asignación de una confianza dada x a un determinado *elemento focal* A del marco, el resto de la confianza no asignada representa la ignorancia o "falta de conocimiento" sobre el grado de importancia de la evidencia en relación al elemento focal considerado[5] y, por tanto, esa confianza no asignada $1 - x$ debe ser asignada al propio

[4] Cuya cardinalidad es, pues, $2^{\#\theta}$, donde $\#\theta$ es la cardinalidad del marco.

[5] En otras palabras, se sabe que la evidencia apoya al elemento focal en un grado x, sin embargo, la confianza no asignada $1 - x$ no sabemos si contribuye o no a A o a cualquier otro subconjunto del marco.

marco de discernimiento porque lo que sí es sabido, por construcción del esquema, es que la solución está en él.

La formulación completa de la aproximación es la siguiente:

$$\begin{aligned} \theta = \{H_1, H_2, \ldots, H_n\} &= \text{marco de discernimiento} \\ A \subset \theta &= \text{elemento focal} \\ e &= \text{evidencia referida a } A \\ m_e(A) &= \text{medida de asignación básica de} \\ &\quad \text{verosimilitud de } A \text{ dado } e \\ e : A &\rightarrow m(A) = x \\ &\quad m(\theta) = 1 - x \\ &\quad m(B) = 0 \qquad \forall B \subset \theta, B \neq \theta, B \neq A \end{aligned}$$

Si el planteamiento fuese probabilístico, la misma evidencia apoyaría al elemento focal A y al complemento del mismo,

$$p(A) = x \rightarrow p(\neg A) = 1 - x$$

y recordemos que éste era uno de los aspectos más débiles de los modelos probabilísticos. De este modo, puede afirmarse que el procedimiento según el cual se maneja la falta de información en la teoría evidencial corrige las carencias de los modelos probabilísticos.

9.1.1. Combinación de Evidencias

Aunque todo lo visto hasta el momento es totalmente correcto, parece claro que en problemas reales las evidencias no vienen solas. Más aún, distintas evidencias no necesariamente tienen por qué referirse a los mismos elementos focales. La cuestión que ocupa esta sección es, pues, la consideración del efecto conjunto de todas las evidencias.

Si dos (o más) fuentes de información proporcionan sendas evidencias relativas a dos elementos focales de un mismo marco de discernimiento, las funciones de asignación básica de verosimilitud se combinan para dar una nueva función de asignación básica de verosimilitud que representa el efecto conjunto de ambas evidencias sobre la intersección de los elementos focales correspondientes:

$$\begin{aligned} e_1 : A_1 \rightarrow m_1(A_1) &= x \\ e_2 : A_2 \rightarrow m_2(A_2) &= y \end{aligned}$$

$$m_{12}(C) = m_1(A_1) \times m_2(A_2) \text{ donde } C = A_1 \cap A_2$$

expresión que puede generalizarse directamente[6] para distintas parejas de elementos focales,

$$m_{12}(C) = \sum_{C = A_i \cap B_j} m_1(A_i) \cdot m_2(B_j)$$

[6]La función de distribución de probabilidad conjunta es, como vemos, el producto.

formulación que coincide con la de asignación de probabilidad a la intersección de dos sucesos independientes en la teoría clásica de la probabilidad, motivo por el que se puede afirmar que la teoría evidencial asume implícitamente independencia entre las evidencias.

Por otra parte, está claro que la primera condición exigida a la función de asignación básica de verosimilitud se cumple,

$$\sum_{C=A_i\cap B_j} m_{12}(C) = 1$$

Sin embargo, puede ocurrir que distintas evidencias "señalen" a elementos focales muy distintos, tanto que no tengan ningún elemento en común, y por tanto su intersección sea nula. Esto introduce una peculiaridad en el modelo:

$$\begin{aligned} e_1 : A_1 \rightarrow m_1(A_1) &= x \text{ con } 0 < x \leq 1 \\ e_2 : A_2 \rightarrow m_2(A_2) &= y \text{ con } 0 < y \leq 1 \end{aligned}$$

$$m_{12}(C) = m_{12}(\varnothing) = m_1(A_1) \times m_2(A_2) = xy \neq 0$$

$$\text{donde } C = A_1 \cap A_2 = \varnothing$$

resultado que contradice la segunda condición exigida a la función básica de asignación de verosimilitud, según la que la solución *tiene* que estar en el marco de discernimiento.

¿Cómo se soluciona esta aparente contradicción? Echando mano de la **normalización**, a fin de conseguir que esta función de asignación de verosimilitud se mantenga dentro de los límites definidos, lo que supone corregir las asignaciones a elementos focales de intersección no nula, de forma que su suma siga siendo la unidad. La nueva expresión para la combinación de evidencias es, finalmente,

$$m_{12}(C) = \frac{\sum\limits_{C=A_i\cap B_j} m_1(A_i)m_2(B_j)}{1 - \sum\limits_{A_i\cap B_j=\varnothing} m_1(A_i)m_2(B_j)} = \frac{\sum\limits_{C=A_i\cap B_j} m_1(A_i)m_2(B_j)}{\sum\limits_{A_i\cap B_j\neq\varnothing} m_1(A_i)m_2(B_j)}$$

La expresión del denominador $K = \sum\limits_{A_i\cap B_j=\varnothing} m_1(A_i)m_2(B_j)$ es una medida de la compatibilidad existente entre las evidencias que están siendo combinadas que se denomina **grado de conflicto**. Así, en la expresión

$$m_{12}(C) = \frac{\sum\limits_{C=A_i\cap B_j} m_1(A_i)m_2(B_j)}{1 - K}$$

el factor $\dfrac{1}{1-K}$ se denomina **factor de normalización**.

Démonos cuenta de que cuando las distintas evidencias señalan a distintos elementos focales entre los que no hay intersecciones nulas, no se puede hablar de evidencias contradictorias, no se dan conflictos, $K = 0$ y la expresión normalizada para $m_{12}(C)$ coincide con la expresión sin normalizar. Por el contrario, cuando las evidencias son totalmente contradictorias (cada evidencia apoya totalmente a una hipótesis -o grupo de - particular y distinta) y todos los elementos focales son disjuntos entre sí, el conflicto es total, $K = 1$ y la combinación de evidencias no está definida.

9.1.2. Credibilidad, Plausibilidad e Intervalo de Confianza

La teoría evidencial permite el seguimiento de la evolución dinámica de la confianza depositada en los subconjuntos del marco de discernimiento a medida que aparecen nuevas evidencias. Para ello se definen dos nuevas medidas, la **credibilidad** y la **plausibilidad** que son indicadores de la mínima y máxima confianza que podemos depositar en un elemento focal dado y se definen:

$$\textbf{Credibilidad} \qquad Cr(A) = \sum_{B \subseteq A} m(B)$$

$$\textbf{Plausibilidad} \qquad Pl(A) = \sum_{B \cap A \neq \varnothing} m(B)$$

La *credibilidad* es una medida de las contribuciones que todos los subconjuntos de A (elemento focal considerado, subconjunto del marco de discernimiento) ejercen sobre el propio A. Por su parte, la *plausibilidad* considera también las contribuciones de otros subconjuntos con intersección no nula, es decir, no sólo tiene en cuenta los subconjuntos del propio elemento focal, sino también todas las contribuciones de todos aquellos subconjuntos que "tienen algo que ver" con dicho elemento A.

Una tercera medida importante es el llamado **intervalo de confianza**, que se construye, para cada elemento focal, a partir de la *credibilidad* y de la *plausibilidad*. Así, en cada nivel del proceso de razonamiento, el intervalo de confianza es el segmento del espacio numérico $[0, 1]$ que tiene como valor mínimo el valor de la credibilidad del elemento focal y como valor máximo el correspondiente valor de la plausibilidad. El *intervalo de confianza* representa la incertidumbre asociada al elemento focal considerado. Cuanto más "cerca" se encuentren los valores de credibilidad y plausibilidad, menos incertidumbre hay respecto a la confianza (más estrecho es el intervalo).

La evolución de la credibilidad en y plausibilidad de un elemento focal a medida que van apareciendo evidencias se comporta de la siguiente manera:

- √ Con una sola evidencia, la credibilidad de un elemento focal coincide exactamente con la medida de asignación básica de verosimilitud, mientras que la plausibilidad es igual a 1.
- √ A medida que aparecen evidencias, la credibilidad se reajusta según las fórmulas que se han indicado, del mismo modo que la plausibilidad.

Se puede demostrar que si A es un elemento focal:

$$\begin{array}{lllll} 0 & \leq & Cr(A) & \leq & 1 \\ 0 & \leq & Pl(A) & \leq & 1 \\ Cr(A) & \leq & Pl(A) & & \\ Cr(A) & \leq & Prob(A) & \leq & Pl(A) \end{array}$$

donde $Prob(A)$ es la probabilidad estadística. Es decir, en el caso en que $Cr(A) = Pl(A)$ la formulación de Dempster y Shafer contiene exactamente a (produce los mismos resultados que) la teoría de la probabilidad.

Credibilidad y plausibilidad se pueden usar, pues, para medir la evolución de la incertidumbre a lo largo del tiempo (con la aparición de evidencias, al reasignar la confianza, se redistribuye la ignorancia entre los diferentes elementos focales del marco de discernimiento). Así,

$$\begin{array}{llllll} \text{Si } Cr(A) & = & 0 \text{ y } Pl(A) & = & 1 & \text{entonces la incertidumbre sobre } A \text{ es total} \\ \text{Si } Cr(A) & = & Pl(A) & = & 1 & \text{entonces } A \text{ es absolutamente cierto} \\ \text{Si } Cr(A) & = & Pl(A) & = & 0 & \text{entonces } A \text{ es absolutamente falso} \\ \text{Si } Cr(A) & = & Pl(A) & = & 0'5 & \text{entonces es absolutamente cierto que no se sabe nada} \end{array}$$

siendo los tres últimos casos ejemplos de *certidumbre completa*.

9.1.3. Casos Particulares de la Teoría Evidencial

La teoría evidencial contiene, bajo ciertos supuestos y en ciertas situaciones, al modelo de los factores de certidumbre de Shortliffe y Buchanan. Más concretamente, la teoría evidencial funcional igual que el modelo de Shortliffe y Buchanan en aquellos casos en los que éste funciona bien (evidencias independientes que apoyan el mismo elemento focal $\equiv$ hipótesis), y lo mejora notablemente cuando el modelo de los factores de certidumbre presenta carencias (casos de dependencia de evidencias).

Capítulo 10

Conjuntos Difusos

En este mundo nada es verdad y nada es mentira, todo es según el color del cristal con que se mira.

Refranero popular español.

Esta es la noción de **conjuntos difusos**, para los que la descripción de objetos y entidades del mundo real debe realizarse según los criterios lingüísticos propios de los seres humanos, que son en su mayoría ambiguas.

Esta ambigüedad es, no obstante, una característica esencial no sólo del lenguaje sino de los procesos de clasificación, establecimiento de taxonomías y jerarquías e incluso de los procesos de razonamiento. Además, la ambigüedad no sólo puede surgir de las definiciones de las cosas, sino que puede ser de carácter subjetivo e incluso debida al contexto.

De modo que los conjuntos ordinarios, en los que un elemento del universo determinado pertenece o no pertenece al conjunto, no nos bastan para representar el conocimiento habitualmente empleado, y mucho menos para razonar con él. Las Matemáticas y la IA, no podían quedar al margen de esta peculiaridad, y en 1965 Lofti Zadeh hizo públicos sus trabajos relacionados con el tema en su famoso artículo "Fuzzy Sets".

10.1. Aspectos Generales de los Conjuntos Difusos

Un conjunto ordinario puede definirse como una colección de elementos[1]. Si un elemento del universo está representado en la colección, el elemento en cuestión pertenece a dicho conjunto. En estos casos se puede decir que el grado de pertenencia de un elemento cualquiera del *universo de discurso* o *referencial* tiene un valor booleano, de forma que:

- $\surd$ si el elemento pertenece al conjunto, el valor booleano es 1
- $\surd$ si el elemento no pertenece al conjunto, el valor booleano es 0

[1]Hay tres maneras de definir un conjunto: *por descripción*, *por enumeración* y, como veremos en este tema, a través de una *función de pertenencia*.

De este modo, puede construirse una función f, para conjuntos ordinarios una función booleana, tal que dado un elemento s del referencial U y dado un subconjunto $A \subset U$:

$$\begin{array}{rcl} f_A(x) & = & 1 \Leftrightarrow x \in A \\ f_A(x) & = & 0 \Leftrightarrow x \notin A \end{array}$$

Ampliaremos ahora la cuestión a ese tipo especial de conjuntos que hemos denominado **conjuntos difusos**. En su caso decíamos que matices de carácter lingüístico, subjetivo, etc. nos impedían establecer con claridad el grado de pertenencia de algunos elementos del referencial al conjunto difuso considerado. Así, habrá elementos del referencial que claramente pertenezcan al conjunto, habrá otros que claramente no pertenezcan y habrá un tercer tipo de elementos que *pertenezcan en cierto grado*.

Debemos entonces considerar que la función f adopta los siguientes valores, dado un elemento x del referencial U y un subconjunto difuso $A \subset U$:

$$\begin{array}{rcl} f_A(x) & = & 1 \Leftrightarrow x \in A \\ f_A(x) & = & 0 \Leftrightarrow x \notin A \\ 0 < f_A(x) < 1 & \Leftrightarrow & x \text{ pertenece en cierto grado a } A \end{array}$$

La función f cuantifica del algún modo el grado de pertenencia de un elemento del referencial al conjunto difuso considerado. Así, un *conjunto difuso* es aquél en el que no existe una frontera clara entre la pertenencia y la no pertenencia de determinados elementos del referencial. Del mismo modo, podemos apreciar que los conjuntos ordinarios son un caso particular de los conjuntos difusos.

Para establecer los "límites difusos" del conjunto correspondiente vamos a necesitar criterios, que casi siempre van a ser arbitrarios, aunque el problema de la definición de criterios para la "fuzzyficación" (*difuminación*) de conjuntos no es trivial.

Por otra parte, la aproximación difusa no es muy natural en términos lingüísticos, con lo cual nos encontramos ante un nuevo problema, el de la clasificación lingüística con conjuntos difusos. Al respecto, la idea básica es que, una vez hemos sido capaces de segmentar el espacio numérico –indicativo de los grados de pertenencia de los elementos del referencial al subconjunto difuso considerado–, debemos segmentar también el espacio lingüístico, estableciendo un conjunto determinado de etiquetas dotadas de contenido semántico, y hacer corresponder a cada etiqueta lingüística un intervalo numérico concreto según un criterio mínimamente razonable. Existen estudios teóricos que tratan de demostrar que la máxima imprecisión lingüística puede conseguirse a través de una escala semántica formada por no más de nueve elementos literales[2].

En resumen, la "fuzzyficación" de un conjunto pasa por los siguientes puntos:

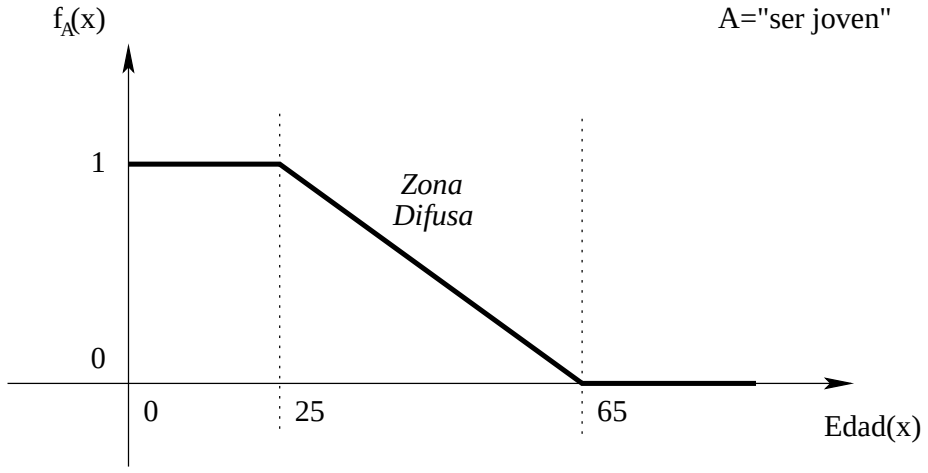
- Identificar las propiedades del conjunto y elegir una de ellas como característica.

[2]Así como en el tema anterior veíamos que el número de etiquetas lingüísticas debía ser un número impar, debido a la escala $(-1, 0, 1)$, en este caso no es necesario porque la gradación es arbitraria y suave.

- Definir criterios de pertenencia.
- Caracterizar el espacio difuso.
- Normalizar lingüísticamente.

Por último, debemos notar una serie de aspectos relacionados con este tratamiento:

- La función $f_A(x)$ definida para la zona difusa no tiene por qué ser lineal, aunque es necesario que sea continua.
- La escala lingüística asociada al espacio numérico, aunque es arbitraria, depende del tipo de clasificación que queremos obtener.
- El número de elementos semánticos de la escala lingüística también es arbitrario.
- En algunos casos es posible definir conjuntos difusos *lingüísticamente complementarios* que hagan más natural la expresión verbal.
- Cualquier conjunto, sea cual sea su naturaleza, es *difuminable*, es decir, se puede establecer una gradación entre los niveles de pertenencia de distintos elementos de un referencial con respecto al conjunto considerado.



$f_A(x) = 1$ para todo x/Edad(x)<25
$f_A(x) = 0$ para todo x/Edad(x)>65
$f_A(x) = \frac{65-Edad(x)}{65-25}$ para todo x/Edad(x) en [25,65]

Figura 10.1: Ejemplo de función de pertenencia a un conjunto difuso.

10.2. Caracterización y Nomenclatura de Conjuntos Difusos

Cualquier conjunto, sea difuso u ordinario, tiene que poder ser descrito de manera conveniente. En el caso de los conjuntos ordinarios, dado que se puede establecer sin ambigüedades la correspondiente relación de pertenencia de los elementos del referencial al conjunto considerado, resulta equivalente caracterizar al conjunto en cuestión en función de su dominio o haciendo explícitos los elementos que lo constituyen.

Por otra parte, ya hemos visto que para cada elemento de un referencial dado, podemos definir una función f (de carácter booleano en el caso de conjuntos ordinarios) tal que a cada elemento del referencial le asignará su valor lógico correspondiente, 0 ó 1, según el elemento en cuestión pertenezca o no al conjunto. De modo que:

Dado un referencial U y un subconjunto del mismo $A \subset U$,

$$\begin{aligned} \exists f_A(x) &= 1 \Leftrightarrow x \in A \\ &= 0 \Leftrightarrow x \notin A \end{aligned}$$

Aplicando este criterio al conjunto ordinario $A = 2, 4, 6, 8$ (conjunto de los naturales pares menores que diez), A estará perfectamente determinado con la expresión:

$$f_A(x) = f_A(1) = 0 + f_A(2) = 1 + f_A(3) = 0 + f_A(4) = 1 + \ldots$$

donde + se lee "y". Otra expresión equivalente, más simplificada, es:

$$f_A(x) = 0/1 + 1/2 + 0/3 + 1/4 + 0/5 + 1/6 + \ldots$$

donde se presentan los valores de la función grado de pertenencia frente a los elementos del referencial considerado.

Por lo tanto, un subconjunto ordinario A de un referencial U puede ser descrito:

- ▷ Implícitamente.
- ▷ Explícitamente.
- ▷ Mediante una $f_A(x)$ booleana $\forall x \in U$.

Por razones obvias, cuando trabajemos con conjuntos difusos preferiremos emplear descripciones implícitas o utilizar las funciones de grado de pertenencia. En este último caso, tendremos en cuenta que se pierde el carácter booleano de la mencionada función:

$$\forall A \subset U \text{ tal que } A \text{ es difuso } \rightarrow \exists f_A(x) / f_A(x) : U \rightarrow [0, 1] \forall x \in U$$

es decir, la función $f_A(x)$ puede tomar cualquier valor en el intervalo $[0, 1]$.

Como veremos más adelante, esta forma de nombrar a los conjuntos difusos nos lleva directamente a establecer que los conjuntos ordinarios son un caso particular de los conjuntos difusos (como ya habíamos mencionado), aunque no tienen la misma estructura algebraica (como demostraremos).

10.3. Estructura Algebraica de los Conjuntos Difusos

Para investigar la estructura algebraica de los conjuntos difusos se tienen que verificar un conjunto de propiedades, que trataremos de establecer y desarrollar a continuación.

Conjunto vacío

Sea un referencial U y sea $Z \subset U$ tal que

$$\exists f_Z(x) : U \to [0,1] \quad \forall x \in U$$

[3] decimos que

$$\mathbf{Z} = \varnothing \Leftrightarrow \mathbf{f_Z(x) = 0} \quad \forall \mathbf{x} \in \mathbf{U}$$

Identidad

Sea un referencial U y sean $Z \subset U$ y $B \subset U$ tales que

$$\exists f_A(x) : U \to [0,1] \quad \forall x \in U$$
$$\exists f_B(x) : U \to [0,1] \quad \forall x \in U$$

decimos que

$$\mathbf{A = B} \Leftrightarrow \mathbf{f_A(x) = f_B(x)} \quad \forall \mathbf{x} \in \mathbf{U}$$

Complementariedad

Sea un referencial U y sea $A \subset U$ tal que

$$\exists f_A(x) : U \to [0,1] \quad \forall x \in U$$

decimos que

$$\mathbf{A' = A_c} \Leftrightarrow \mathbf{f'_A(x) = 1 - f_A(x)} \quad \forall \mathbf{x} \in \mathbf{U}$$

Obviamente, $f'_A(x) : U \to [0,1]$.

[3] Expresión que define a Z como conjunto difuso (*p.examen*).

Inclusión

Sea un referencial U y sean $A \subset U$ y $B \subset U$ tales que

$$\begin{array}{ll} \exists f_A(x) : U \to [0,1] & \forall x \in U \\ \exists f_B(x) : U \to [0,1] & \forall x \in U \end{array}$$

decimos que

$$\mathbf{B \subset A \Leftrightarrow f_B(x) \leq f_A(x) \quad \forall x \in U}$$

caracterización que es totalmente análoga a la que obtendríamos si considerásemos conjuntos ordinarios y los describiésemos con notación difusa.

Unión

Sea un referencial U y sean $A \subset U$, $B \subset U$ y $C \subset U$ tales que

$$\begin{array}{ll} \exists f_A(x) : U \to [0,1] & \forall x \in U \\ \exists f_B(x) : U \to [0,1] & \forall x \in U \\ \exists f_C(x) : U \to [0,1] & \forall x \in U \end{array}$$

decimos que

$$\mathbf{C = A \cup B \Leftrightarrow f_C(x) = max\{f_A(x), f_B(x)\} \quad \forall x \in U}$$

algo que es intuitivo en conjuntos ordinarios pero no tanto con conjuntos difusos. La unión se puede denotar también como $f_A(x) or f_B(x) = f_A(x) \vee f_B(x)$ o simplemente $f_A or f_B$ y puede demostrarse que tiene la propiedad asociativa.

Intersección

Sea un referencial U y sean $A \subset U$, $B \subset U$ y $C \subset U$ tales que

$$\begin{array}{ll} \exists f_A(x) : U \to [0,1] & \forall x \in U \\ \exists f_B(x) : U \to [0,1] & \forall x \in U \\ \exists f_C(x) : U \to [0,1] & \forall x \in U \end{array}$$

decimos que

$$\mathbf{C = A \cap B \Leftrightarrow f_C(x) = min\{f_A(x), f_B(x)\} \quad \forall x \in U}$$

La intersección se puede denotar también como $f_A(x) and f_B(x) = f_A(x) \wedge f_B(x)$ o simplemente $f_A and f_B$ y puede demostrarse que tiene la propiedad asociativa.

Leyes de DeMorgan

Sea un referencial U y sean $A \subset U$ y $B \subset U$ tales que

$$\begin{array}{ll} \exists f_A(x) : U \to [0,1] & \forall x \in U \\ \exists f_B(x) : U \to [0,1] & \forall x \in U \end{array}$$

Las leyes de DeMorgan establecen (y se puede demostrar) que:

- El complementario de la unión equivale a la intersección de los complementarios:
$$(A \cup B)' = A' \cap B'$$
- El complementario de la intersección equivale a la unión de los complementarios:
$$(A \cap B)' = A' \cup B'$$

Leyes Distributivas

Los conjuntos difusos verifican también las leyes distributivas:

- Distributividad de la intersección respecto de la unión:
Dado un referencial U y dados $A \subset U$, $B \subset U$ y $C \subset U$ tales que
$$\begin{array}{ll} \exists f_A(x) : U \to [0,1] & \forall x \in U \\ \exists f_B(x) : U \to [0,1] & \forall x \in U \\ \exists f_C(x) : U \to [0,1] & \forall x \in U \end{array}$$
se cumple que
$$C \cap (A \cup B) = (C \cap A) \cup (C \cap B)$$
- Distributividad de la unión respecto de la intersección:
Dado un referencial U y dados $A \subset U$, $B \subset U$ y $C \subset U$ tales que
$$\begin{array}{ll} \exists f_A(x) : U \to [0,1] & \forall x \in U \\ \exists f_B(x) : U \to [0,1] & \forall x \in U \\ \exists f_C(x) : U \to [0,1] & \forall x \in U \end{array}$$
se cumple que
$$C \cup (A \cap B) = (C \cup A) \cap (C \cup B)$$

Según lo visto hasta ahora todo parece indicar que los conjuntos difusos tienen estructura de álgebra de Boole; sin embargo, hay dos leyes del álgebra de Boole que los conjuntos difusos no satisfacen, que son el **principio de no contradicción** y la **ley del tercero excluído**:

Ley del Tercero Excluído

Dado un referencial U y dado $A \subset U$, donde A es ordinario, se cumple que
$$A \cup A' = U$$
Sin embargo, si A es un conjunto difuso, es decir, $\exists f_A(x) : U \to [0,1] \quad \forall x \in U$,
$$A \cup A' \to f_A \cup f_{A'} = max\{f_A(x), f_{A'}(x)\} = max\{f_A(x), 1 - f_A(x)\} \quad \forall x \in U$$
que es siempre mayor o igual a $\frac{1}{2}$ pero no necesariamente 1.

Principio de No Contradicción

Dado un referencial U y dado $A \subset U$, donde A es ordinario, se cumple que

$$A \cap A' = \varnothing$$

Sin embargo, si A es un conjunto difuso, es decir, $\exists f_A(x) : U \to [0,1] \quad \forall x \in U$,

$$A \cap A' \to f_A \cap f_{A'} = min\{f_A(x), f_{A'}(x)\} = min\{f_A(x), 1 - f_A(x)\} \quad \forall x \in U$$

que es siempre menor o igual a $\frac{1}{2}$ pero no necesariamente 0.

10.4. Operaciones Algebraicas con Conjuntos Difusos

El desarrollo efectuado hasta ahora nos permite describir algunas operaciones algebraicas que podemos realizar con conjuntos difusos. La descripción de tales operaciones se realizará a partir de las correspondientes funciones de grado de pertenencia.

Producto

Sea un referencial U y sean $A \subset U$ y $B \subset U$ tales que

$$\exists f_A(x) : U \to [0,1] \quad \forall x \in U$$
$$\exists f_B(x) : U \to [0,1] \quad \forall x \in U$$

definimos

$$\mathbf{A \times B \to f_{AB}(x) = f_A(x) \cdot f_B(x) \quad \forall x \in U}$$

Se verifica que el producto de conjuntos ordinarios coincide con su intersección, mientras que en el caso de los conjuntos difusos se observa que

$$f_{AB}(x) \leq f_{A \cap B}(x) \quad \forall x \in U$$

es decir, el producto es más restrictivo que la intersección.

Suma y Suma Acotada

Sea un referencial U y sean $A \subset U$ y $B \subset U$ tales que

$$\exists f_A(x) : U \to [0,1] \quad \forall x \in U$$
$$\exists f_B(x) : U \to [0,1] \quad \forall x \in U$$

definimos

$$\mathbf{A + B \to f_{A+B}(x) = f_A(x) + f_B(x) \quad \forall x \in U}$$

La suma de conjuntos difusos sólo está definida cuando $f_A(x) + f_B(x) \leq 1 \quad \forall x \in U$. Para evitar este problema se define la suma acotada

$$\mathbf{A| + |B \to f_{A|+|B}(x) = min\{1, f_A(x) + f_B(x)\} \quad \forall x \in U}$$

Diferencia y Diferencia Absoluta

Sea un referencial U y sean $A \subset U$ y $B \subset U$ tales que

$$\exists f_A(x) : U \to [0,1] \quad \forall x \in U$$
$$\exists f_B(x) : U \to [0,1] \quad \forall x \in U$$

definimos

$$\mathbf{A - B \to f_{A-B}(x) = f_A(x) - f_B(x) \quad \forall x \in U}$$

La diferencia de conjuntos difusos sólo está definida cuando $f_B(x) \leq f_A(x) \quad \forall x \in U$, es decir, sólo se puede establecer cuando $B \subset A$. Para evitar este problema se define la diferencia absoluta

$$\mathbf{|A - B| \to f_{|A-B|}(x) = |f_A(x) - f_B(x)| \quad \forall x \in U}$$

Núcleo

Sea un referencial U y sea $A \subset U$ tal que

$$\exists f_A(x) : U \to [0,1] \quad \forall x \in U$$

definimos

$$\mathbf{nucleo(A) = N_A = \{x \in U / f_A(x) = 1\}}$$

Un conjunto difuso se dice *normalizado* si tiene núcleo.

Relación difusa

Dado un referencial U definimos una *relación difusa* de orden n en U como un conjunto difuso A en el espacio $\underbrace{U \times U \times \ldots \times U}_{n \text{ veces}}$ caracterizado por una función de grado de pertenencia del tipo

$$f_A(x_1, x_2, \ldots, x_n) \quad \forall x \in U$$

10.5. Representación del Conocimiento y Razonamiento Difuso

Iniciaremos ahora una aproximación a la representación del conocimiento y el razonamiento difusos. En las frases del lenguaje natural podemos reconocer predicados difusos, cuantificadores difusos e incluso probabilidades difusas (no numéricas). Las aproximaciones más convencionales usualmente empleadas para representar conocimiento (modelos basados en lógica de primer orden o en las teorías clásicas de la probabilidad) carecen de medios para representar eficazmente el significado de conceptos difusos y no permiten por tanto manipular correctamente el conocimiento de sentido común. La causa evidente es fundamentalmente que el conocimiento derivado del sentido común es léxicamente impreciso y de naturaleza no categórica.

Las características estudiadas de los conjuntos difusos nos dan pistas sobre la que sería una manera más adecuada de proceder, si lo que queremos es aplicar esquemas de representación del conocimiento y modelos de razonamiento basados en lógica difusa:

- √ En lógica difusa el razonamiento categórico es un caso particular del razonamiento aproximado.
- √ En lógica difusa todo es una cuestión de grado.
- √ Cualquier sistema lógico puede ser "fuzzyficado".
- √ En lógica difusa el conocimiento debe ser interpretado como una colección de *restricciones difusas* que operan sobre una colección de variables.
- √ En lógica difusa los problemas de razonamiento (y por tanto los procesos inferenciales) deben interpretarse como *propagaciones* de las restricciones difusas.

La cuestión es: ¿cómo podríamos representar en un sistema difuso una declaración del tipo "*Si x es A, entonces y es B*", donde A es un subconjunto difuso de un referencial U y B es un subconjunto difuso de un referencial V (que puede ser igual o distinto a U) y $x \in U$, $y \in V$? La respuesta no es única y varios autores proponen distintas soluciones. Zadeh, por su parte, sugiere que la función de grado de pertenencia de una declaración de este tipo puede calcularse:

$$f_A \rightarrow B(x,y) = A'| + |B = min\{1, 1 - f_A(x) + f_B(y)\} \quad x \in U, y \in V$$

Así se introduce el mecanismo de inferencia conocido como *modus ponens*, que generalizado para conjuntos difusos se puede representar:

```
A -> B
a
--------
      b
```

donde `a` se parece a `A` pero no es `A` y `b` se parece a `B` pero no es `B`. Así, la expresión para calcular $f_B(y)$ es:

$$f_b(y) = sup_V[A'| + |B] \cap a \qquad A \subset U, a \subset U, B \subset V, b \subset V$$

por lo que $f_b(y) = sup_V[min\{min\{1, 1 - f_A(x) + f_B(y)\}, f_a(x)\}] \quad x \in U, y \in V$.

No obstante, el modus ponens es la única diferencia en el razonamiento de los sistemas difusos frente al razonamiento en sistemas más clásicos y convencionales.

Certeza

En sistemas que utilizan lógica bivalente la verdad de una declaración sólo puede tener dos valores: cierta o falsa. Por el contrario, en sistemas multivaluados la verdad de una declaración puede ser un elemento de un conjunto finito, un intervalo o un álgebra de Boole. Particularmente, en lógica difusa la verdad de una declaración puede ser un subconjunto difuso parcialmente ordenado, pero normalmente se asume la existencia de un subconjunto difuso del intervalo $[0, 1]$ o, dicho de otro modo, un punto de dicho intervalo. Así, los denominados valores lingüísticos de la verdad de una declaración pueden expresarse por medio de etiquetas del tipo *cierto*, *muy cierto*, *no exactamente cierto*, ..., correspondientes a subconjuntos difusos del mencionado intervalo.

Predicados

En sistemas bivalentes los predicados son categóricos, pero en sistemas difusos los predicados son, precisamente, difusos.

Modificadores

En sistemas clásicos el único modificador realmente utilizado es la negación `not`. En sistemas difusos hay una gran variedad de modificadores (*muy*, *más*, *bastante*,...) que son esenciales para generar los valores apropiados de las variables lingüísticas involucradas en un proceso.

Cuantificadores

En los sistemas clásicos hay únicamente dos cuantificadores, el universal y el existencial. En los sistemas difusos, por el contrario, encontramos una gran variedad de cuantificadores (*pocos*, *bastantes*, *normalmente*, *la mayoría*,...).

Probabilidades

En los sistemas lógicos clásicos la probabilidad es numérica. En los sistemas difusos, la probabilidad se expresa por medio de etiquetas lingüísticas (probabilidades difusas: *plausible*, *poco probable*, *alrededor de*,...). El manejo de tales probabilidades difusas debe efectuarse a través de la aritmética difusa.

Posibilidades

A diferencia de en los sistemas lógicos clásicos, el concepto de posibilidad en los sistemas difusos no es bivalente. De hecho, al igual que con las probabilidades, las posibilidades pueden ser tratadas como variables lingüísticas que adoptan valores del tipo *casi imposible*, *bastante posible*,...

Modos de razonamiento: Razonamiento Categórico

El Razonamiento Categórico utiliza declaraciones difusas, pero no emplea ni cuantificadores difusos ni probabilidades difusas (son predicados categóricos sobre declaraciones difusas, cuyos predicados de conclusión son difusos, conjunción de las premisas anteriores).

Modos de razonamiento: Razonamiento Silogístico

El Razonamiento Silogístico produce inferencias con premisas que incorporan cuantificadores difusos (y utilizan por tanto aritmética difusa para la obtención de sus conclusiones).

Modos de razonamiento: Razonamiento Disposicional

En el Razonamiento Disposicional las premisas son disposiciones y la conclusión obtenida es una máxima que debe interpretarse como un mandato disposicional.

Modos de razonamiento: Razonamiento Cualitativo

Por último, el Razonamiento Cualitativo se define en sistemas difusos como un modo de razonamiento en el cual las relaciones de entrada/salida de un sistema se representan por medio de una colección de reglas difusas (tipo `IF-THEN`) en las que los antecedentes y los consecuentes incluyen variables lingüísticas. Este tipo de razonamiento es el empleado habitualmente en las aplicaciones de la lógica difusa al análisis de sistemas y control de procesos.

Actualmente la aplicación de los conjuntos difusos a los sistemas inteligentes es un tema de gran interés en investigación. De todas formas, aunque las bases teóricas del formalismo difuso están bastante claras, su aplicación a sistemas de naturaleza inferencial encuentra problemas que hoy en día siguen sin estar resueltos. Sí parece, no obstante, que los sistemas difusos aplicados a problemas de control están proporcionando soluciones alternativas y de gran brillantez y elegancia frente a planteamientos más tradicionales.

Capítulo 11

Introducción a la Ingeniería del Conocimiento

Como *ciencia*, hemos visto que la IA trata de desarrollar el vocabulario y los conceptos que permiten ayudar a comprender, y en ocasiones a reproducir, comportamiento inteligente. Como *ingeniería*, trata de definir y formalizar un conjunto de métodos que nos permitan adquirir conocimiento de algo nivel y representarlo según un esquema computacionalmente eficaz, para resolver problemas difíciles en dominios de aplicación concretos. Es decir, la IA como *ciencia* desarrolla modelos, y como **ingeniería del conocimiento** los aplica para tratar de resolver problemas intelectualmente complicados. Los programas resultantes de la aplicación de técnicas de **ingeniería del conocimiento** se denominan **Sistemas Expertos**.

11.1. Características Generales de los Sistemas Expertos

Los **Sistemas Expertos** son programas inteligentes diseñados para asistir a los expertos humanos en dominios del mundo real, limitados en extensión, pero intelectualmente difíciles. Tratan de modelizar en un programa el conocimiento y el modo de razonar de los expertos humanos, por lo que no tienen por qué proponer las mismas soluciones a los mismos problemas ni se les debe exigir que proporcionen la "mejor solución", sino que basta con que sea aceptable.

Desde una perspectiva estructural, los *Sistemas Expertos* no sólo representan al dominio que tratan de modelizar, sino que también deben conservar representaciones de su propia estructura interna y su funcionamiento. Esta última característica, el **autoconocimiento**, es la que permite a los sistemas expertos justificar sus conclusiones, explicar sus procesos de razonamiento e incrementar dinámicamente el conocimiento que poseen.

Para conseguir avances en estos puntos, la *Ingeniería del Conocimiento* ha sugerido y sugiere arquitecturas que separan claramente los conocimientos del dominio de los mecanismos de inferencia y control.

Otro aspecto relativo a los *Sistemas Expertos* es la importancia decisiva de la experiencia a la hora de resolver un problema. Esta experiencia, el *conocimiento heurístico*, está íntimamente relacionada con los conocimientos y métodos de actuación de un verdadero experto humano. El grado de experiencia de un profesional cualquiera en un dominio concreto suele repercutir en la capacidad del individuo en cuestión para resolver ciertos problemas en los que otros de la misma profesión fracasarían o no tendrían tanto éxito.

En cualquier caso, al margen de estas consideraciones hay que destacar el papel preponderante de los dominios de aplicación y de las tareas para las que se precisan conocimientos y experiencia. No todos los dominios ni todas las tareas son igualmente apropiadas para la construcción de Sistemas Expertos[1]. Muchos problemas pueden ser resueltos utilizando técnicas de programación convencional[2], mientras que otros requieren la aplicación de técnicas de Ingeniería del Conocimiento[3].

Atendiendo a los diferentes dominios que un ingeniero del Conocimiento puede encontrarse y considerando idóneas aquellas tareas cuya ejecución y/o resolución se requiere experiencia, en los términos anteriormente expuestos, podemos señalar algunos problemas tipo para los cuales podría ser deseable, e incluso conveniente, el diseño y desarrollo de un sistema experto:

- Interpretación de Información.
- Predicción.
- Pronóstico y Prevención.
- Diseño.
- Planificación.
- Monitorización y Supervisión.
- Ayuda a la decisión.
- Enseñanza asistida por ordenador.
- Control.
- Aprendizaje.

Estas suelen ser tareas típicas que aconsejan el empleo de técnicas de IC para construir programas inteligentes, en casi cualquier dominio de aplicación. Es decir, los SE no tienen sentido en dominios deterministas, y lo adquieren en los que es necesaria cierta experiencia[4].

[1] En adelante, *SE*.

[2] Y si es así, es preferible, ya que la informática convencional es más rápida en ejecución y desarrollo que la informática inteligente.

[3] En adelante, *IC*.

[4] ¿Sería deseable un SE en un sistema diseñado para interpretar información? Sí, desde el momento en que usar información supone usar *conocimiento*.

11.2. Análisis de la Viabilidad de un Sistema Experto

Los SE contienen información estructurada, razonada y adaptable a cambios sucesivos. Por otra parte, los expertos humanos en un dominio concreto son escasos, difíciles de encontrar y caros de mantener. Podemos considerar que los SE *popularizan* el conocimiento de los expertos humanos y reducen el coste de formación de nuevos expertos. Pero la construcción de un SE realmente útil es un proceso largo y costoso, de modo que *¿cuándo es realmente viable la construcción de un SE?*

D.A.Waterman trata de responder a esta pregunta proponiendo una metodología de diseño basada en el estudio de cuatro características esenciales: *Justificación*, *Posibilidad*, *Adecuación* y *Éxito*:

```
IF:   La construcción del SE está justificada
AND:  La construcción del SE es posible
AND:  La construcción del SE es adecuada
AND:  Hay ciertas garantías de éxito tras la construcción del SE

THEN: La construcción del SE es viable
```

Este análisis de viabilidad debe ser previo a cualquier intento de desarrollo y trata de identificar dominios, problemas y tareas en los que verdaderamente merece la pena intentar la construcción de un SE. Ahora bien, ¿cuándo podemos decir que un SE está justificado, su construcción es posible, adecuada, o podemos albergar razonables esperanzas de éxito tras su implementación? Seguimos con la metodología de Waterman:

Justificación

Se define el criterio de *justificación* basándose en realidades de tipo social, económico o coyuntural:

```
IF:   Hay necesidad de experiencia en un entorno hostil
OR:   Existe una verdadera carencia de experiencia humana
OR:   Se necesita experiencia simultánea en distintos lugares
OR:   Se ha detectado pérdida de experiencia humana (en un dominio concreto)
OR:   Hay una alta tasa de recuperación de la inversión
OR:   No hay soluciones alternativas
OR:   Un enfoque de programación convencional no es satisfactorio

THEN: El desarrollo de un SE está justificado
```

Como se puede ver, el análisis de viabilidad introduce matices subjetivos, algo que es una constante en toda la metodología, siendo labor del ingeniero de Conocimiento decidir cuándo y cuándo no se verifican las premisas correspondientes.

Posibilidad

El criterio de *posibilidad* hace referencia a algunos de los aspectos más pragmáticos en el desarrollo de un SE.

```
IF:   Existen varios expertos cooperativos
AND:  Los expertos logran ponerse de acuerdo
AND:  Los expertos son capaces de articular sus métodos
        y procedimientos de trabajo
AND:  Las tareas no son excesívamente difíciles
AND:  Las tareas están suficientemente estructuradas
AND:  El sentido común no es determinante
AND:  Se dispone de un número suficiente de casos relevantes para la
        verificación y posterior validación del producto
AND:  Las tareas no son exclusivamente de carácter teórico

THEN: El desarrollo de un SE es posible
```

Esta fase trata de averiguar si somos o no capaces de articular las fuentes de conocimiento, las tareas que hay que resolver y los medios disponibles.

Adecuación

El criterio de *adecuación* implica investigar tres frentes, todos ellos vinculados al tipo de problema que queremos resolver:

```
IF:   La NATURALEZA del problema aconseja el desarrollo de un SE
AND:  La COMPLEJIDAD del problema aconseja el desarrollo de un SE
AND:  El ALCANCE del problema aconseja el desarrollo de un SE

THEN: El desarrollo de un SE es adecuado
```

Cada una de estas tres características (*Naturaleza*, *Complejidad* y *Alcance*) se deben investigar por separado:

```
IF:   El producto desarrollado cubre necesidades a largo plazo
        (su dominio es estable, no cambiante)
AND:  Las tareas involucradas no requieren investigación básica
        (globalmente)
AND:  Las tareas requieren manipulación simbólica
AND:  Las tareas requieren soluciones heurísticas

THEN: La naturaleza del problema aconseja la construcción de un SE
```

```
IF:   Las tareas no son demasiado fáciles
AND:  El conocimiento necesario aconseja la definición de más de una base
        de conocimientos
AND:  Es posible planificar efectos

THEN: La complejidad del problema aconseja la construcción de un SE


IF:   Las tareas tienen valor práctico
AND:  Las tareas tienen un tamaño manejable
AND:  No es previsible que el producto quede inmediatamente obsoleto

THEN: El alcance del problema aconseja la construcción de un SE
```

Éxito

El estudio del *éxito* debe concentrarse casi exclusivamente en aspectos económicos, políticos y de mercado. Este hecho marca una de las diferencias entre la IA como ciencia y la IA como ingeniería: mientras que la ciencia se preocupa de la creación y difusión de conocimientos, la ingeniería se preocupa de construir productos rentables.

```
IF:   Se efectúa una transferencia de tecnología adecuada
        (se patenta y comercializa)
AND:  Los directivos están mentalizados y tienen perspectivas realistas
AND:  Hay cambios mínimios en los procedimientos habituales (usabilidad)
AND:  Los usuarios finales no rechazan categóricamente
        la tecnología de los SE
AND:  Los resultados no dependen de vaivenes políticos
AND:  El dominio es relativamente estable
AND:  Los objetivos están adecuadamente definidos

THEN: Hay ciertas garantías de éxito tras la construcción de un SE
```

Todo este análisis de viabilidad puede ser representado mediante un circuito inferencial muy sencillo. En cualquier caso, Waterman no dice qué sucede si alguno de los requisitos del análisis de viabilidad no es satisfecho[5] (es decir, en ese caso la viabilidad no está definida). Este modelo, además, es booleano, pero podría hacerse con cualquier método de razonamiento impreciso de los que hemos estudiado.

11.3. Organización General de un Sistema Experto

Una vez investigada la viabilidad del SE conviene recordar brevemente la arquitectura típica de este tipo de sistemas (sistemas de producción), que consta de los siguientes grandes bloques:

[5]Curiosidad: "¿Es viable un SE para análisis de viabilidad de SE?" Obviamente no, pues la complejidad no lo justifica, este análisis es muy sencillo.

- Bases de conocimientos
- Motor de inferencias
- Memoria activa[6]

A estos tres bloques añadiremos un cuarto, el de *Interfaces con el Usuario*, del que en gran medida puede llegar a depender el éxito de nuestro sistema experto.

11.3.1. Bases de Conocimientos

En las bases de conocimientos debemos considerar la inclusión y articulación de tres módulos que incorporan diferentes tipos de conocimiento:

- *Conocimiento declarativo o descriptivo*, que se refiere a los elementos descriptivos del dominio de discurso, contemplados desde una perspectiva estática:
  - Objetos del universo
  - Relaciones estáticas entre objetos
  - Definiciones
  - Vocabulario
  - Hechos
  - Hipótesis, suposiciones, restricciones y taxonomías
- *Conocimiento operativo o de acción*, integrado por entidades que describen el dominio de discurso desde una perspectiva dinámica:
  - Procesos y *demons*
  - Reglas
  - Heurísticas
  - Ejemplos
- *Metaconocimiento* (conocimiento sobre conocimiento), tipo de conocimiento operativo que, a nivel local, permite controlar el funcionamiento del sistema. Un ejemplo típico son las "metarreglas"[7]

11.3.2. Motor de Inferencias

El motor de inferencias consta básicamente de un *intérprete* y de un *módulo de control* (intérprete+estrategia). Más específicamente, deberemos definir e implementar en el motor de inferencias[8] estructuras que nos permitan ejecutar al menos algunas de las siguientes tareas:

[6]Contiene la descripción completa del estado actual del sistema durante un proceso de ejecución, por lo que durante la fase de desarrollo podemos prescindir de su descripción pormenorizada.

[7]Elementos de control con estructura de reglas que se utilizan para regular procesos, por ejemplo: `IF: Hay más de una regla activada THEN: Ejecutar primero la que haya sido más recientemente activada`.

[8]En realidad no se hace nunca, hay motores de inferencias comerciales muy buenos.

- ⋆ Definición del tipo de encadenamiento y gestión del mismo
- ⋆ Mecanismos de unificación, emparejamiento e interpretación
- ⋆ Gestión de prioridades, agendas y pizarras
- ⋆ Modelos y esquemas de razonamiento del sistema
- ⋆ Cálculos

11.3.3. Interfaces

Este es el tercer gran módulo al que debemos prestar atención durante la fase de IC. Su misión es permitir que el SE interaccione con el usuario y con el mundo exterior. Dentro de este módulo podemos distinguir tres subsistemas genéricos:

- ∗ *Subsistemas de usuario*, que permiten la interacción uni o bidireccional del SE con el usuario o con los dispositivos de entrada de información y datos. Típicamente esta interacción se consigue a través de menús, gráficos, rutinas de adquisición de datos que actúan sobre sistemas conectados a instrumentos, accesos a bases de datos, etc.
- ∗ *Subsistemas de explicación*, que facilitan el seguimiento de los procesos inferenciales realizados por el sistema. Este seguimiento suele ser requerido en momentos precisos mientras tiene lugar un proceso consultivo determinado, aunque también puede ser utilizado para la validación y depuración del producto desarrollado. Un subsistema típico de explicación debe ser capaz de responder adecuadamente a cuestiones: "¿cómo?" (visualizando las reglas de producción), "¿por qué?" (visualizando la pila de objetivos y reglas activadas) y "justifícate" (visualizando las reglas desde los datos hasta el final). Los subsistemas de explicación utilizan con profusión el *autoconocimiento* del sistema.
- ∗ *Subsistemas de actualización del conocimiento*, que deben posibilitar la adquisición de nuevo conocimiento y la constante actualización del conocimiento ya existente en el sistema. Típicamente esta tarea se facilita considerablemente con el empleo de procesadores de lenguaje natural o la utilización de herramientas de IC (shells).

11.4. Fases de la Adquisición del Conocimiento

Ya hemos mencionado en alguna ocasión que no podemos considerar que algo muestre un comportamiento inteligente si no utiliza de manera *eficaz* y *eficiente* un conjunto mínimo de conocimientos. En este sentido, los problemas vistos hasta ahora son útiles para comprender los principios básicos de la IA, pero no nos permiten formarnos una idea de lo que realmente subyace tras los SE. Disponemos de una estructura, una arquitectura y unas técnicas de búsqueda de soluciones, de representación del conocimiento, pero falta todavía dotar a estos elementos de contenido, es decir, de **conocimiento**.

La adquisición del conocimiento, en el ámbito estricto de la IC, es sin duda uno de los cuellos de botella de la IA, hasta el punto que es su problema actual, dado el desarrollo de las herramientas de que se dispone hoy en día para la construcción de SE.

Dado un dominio cualquiera, no podemos pretender resolver el problema de la adquisición del conocimiento sentando a un grupo de expertos y dialogando con ellos. Hay que tener muy presente que el objetivo final es la creación de un modelo computacional cualitativo de comportamiento inteligente, en un dominio de aplicación concreto. Para ello, como paso previo, hay que unificar terminología:

Dominio Área de aplicación sobre la que queremos construir nuestro sistema.

Tareas Problemas que se van a presentar en el dominio y que tendremos que resolver.

Métodos Estrategias y/o procedimientos de resolución de los problemas planteados.

Sobre todos y cada uno de los puntos anteriores tendremos que definir nuestra metodología de adquisición del conocimiento teniendo en cuenta que modelizar un problema implica extraer conocimiento del mundo exterior, articularlo, estructurarlo y traducirlo computacionalmente. No se modeliza un problema simplemente transfiriendo experiencia y conocimientos. Además, las bases de conocimientos son modelos del dominio de aplicación; en ellas se incluyen entidades relevantes, procesos y estrategias de resolución de problemas. Y por último, el conocimiento de los expertos, tal y como ellos nos lo comunican, es incompleto, aproximado y poco operativo computacionalmente hablando.

En base a estos argumentos, nuestra labor de adquisición del conocimiento debe seguir una metodología concreta, que observe las siguientes cinco fases:

- √ **Conceptualización**
- √ **Formalización de conceptos**
- √ **Elicitación**
- √ **Operacionalización**
- √ **Verificación y revisión**

11.4.1. Conceptualización

La **conceptualización** se refiere al dominio de aplicación y a las tareas que pretendemos resolver. Esta fase permite identificar los elementos clave, las relaciones, los procesos y otras entidades del dominio que son relevantes en la construcción de un SE concreto. Para conceptualizar correctamente un dominio tendremos que efectuar un análisis completo y detallado de las tareas identificadas y tratar de caracterizar qué es lo que hay que resolver, cuáles son las entradas disponibles, cuáles son las salidas deseadas y qué tipo de información necesitamos para resolver los problemas. Además, tendremos que comprender, pensando en su posterior articulación, los métodos y técnicas apropiadas para la resolución de los problemas del dominio.

11.4.2. Formalización

Podemos definir la **formalización** como el proceso de construccción de representaciones simbólicas que nos permitan "traducir" los resultados de la conceptualización, de carácter abstracto, en algo que pueda ser implementado en una máquina. Así, la formalización de conceptos implica asimilar las entidades clave, las relaciones, los métodos, etc. y encontrar un procedimiento estructurado o lógico para su posterior representación computacional (frames, reglas, predicados, funciones,...). En definitiva, se trata de decidir qué tipo de estructura es más útil para representar cada una de las entidades relevantes identificadas tras el proceso de conceptualización.

11.4.3. Elicitación

La **elicitación** es un proceso de extracción del conocimiento de los expertos humanos que se realiza de una forma estructurada y consistente con el proceso de conceptualización. Supone, en primer lugar, diseñar algún tipo de estrategia estructurada de interlocución con los expertos. Ello puede obligarnos a identificar previamente distintas categorías de información y establecer unos formatos adecuados a la lógica de los procesos con los que nos enfrentamos. A continuación, procede el diseño y materialización de alguna estructura física (formularios, interfaces,...) con la que llevar a cabo la elicitación del conocimiento. Finalmente, esta fase concluye con la traducción de la información adquirida en una representación estructurada o formal.

11.4.4. Operacionalización

En esta fase se trata de hacer computacionalmente operativos los conocimientos previamente elicitados. Para ello trataremos de encontrar o construir los procedimientos de representación más apropiados que permitan al conocimiento elicitado ejecutar las tareas deseadas. Ello supone encontrar el procedimiento computacional que mejor simule las estrategias de resolución descritas por los expertos, seleccionar, adaptar o desarrollar intérpretes y construir prototipos y realizar simulaciones que incluyan diversos modos de interacción con el usuario.

11.4.5. Verificación y revisión

Es la última fase del proceso y con ella se pretende comprobar el funcionamiento de las estructuras implementadas y, si procede, efectuar las correcciones oportunas. Desgraciadamente, constataremos la necesidad de efectuar varias reimplementaciones a medida que el proyecto avanza, según vamos analizando los resultados de las sucesivas verificaciones y revisiones, por diversas razones: el experto suele cambiar de idea conforme el sistema evoluciona, acaba por acostumbrarse al prototipo y comienza a exigir más cosas, el ingeniero de conocimiento va aprendiendo sobre el dominio de aplicación y él mismo sugiere posibles mejoras del sistema, y se familiariza con el entorno de trabajo del sistema y suele proponer modificaciones acerca de la integración final del sistema.

En cualquier caso, la verificación y revisión del sistema inteligente en desarrollo también debe seguir unas pautas bien definidas. Un procedimiento adecuado para llevarla a cabo podría ser:

- construir un primer prototipo que opere siempre con un mismo conjunto de entradas y refinarlo hasta obtener la respuesta deseada
- realizar una primera revisión en la que se le permita al usuario introducir nuevos datos, lo que supone dinamizar el sistema en desarrollo y permite detectar errores en los procesos de razonamiento implementados
- realizar sucesivas revisiones según el procedimiento anterior hasta conseguir un sistema optimizado que seea capaz de trabajar con información y datos reales

11.5. Técnicas de Extracción del Conocimiento

Mencionaremos ahora brevemente algunas de las distintas técnicas que se pueden emplear en el proceso de **extracción del conocimiento**.

Observación directa

La **observación directa** es en realidad una técnica de extracción del conocimiento previa al diseño real del sistema inteligente. Consiste simplemente en la observación pasiva del modo en que un experto se enfrenta con los problemas del dominio de aplicación, con el objetivo de familiarizar al ingeniero de conocimiento con éste y su entorno (se relaciona con la fase de *Conceptualización*).

Disección del problema

Tras la observación directa, procede la discusión informal con los expertos sobre un conjunto de problemas representativos del dominio, lo que se denomina **disección del problema**. El objetivo buscado es averiguar de qué manera los expertos tienden a organizar sus conocimientos, cómo representan mentalmente sus conceptos y sus ideas, cómo analizan la información inconsistente, inexacta o imprecisa. Pueden formularse las preguntas siguientes:

- ⋆ ¿qué características diferenciales tiene este problema concreto frente a otros del dominio?
- ⋆ ¿qué tipo de información es precisa y qué datos son relevantes en la resolución del problema?
- ⋆ ¿qué tipo de soluciones son adecuadas?
- ⋆ ¿podemos reducir el problema planteado a subproblemas no interactivos (independientes[9])?

[9]Tratar con subproblemas no independientes puede dar lugar a la aparición de información redundante, conflictiva,...

- ⋆ ¿qué tipo de conocimientos se necesitan para resolver el problema?
- ⋆ ¿qué elementos básicos deben incluirse en una explicación correcta, adecuada y suficientemente informativa?

El resultado de una disección del problema bien hecha suele ser la aparición de nuevos términos, conceptos y relaciones.

Descripción del problema

Supone describir problemas típicos relacionados con cada categoría importante de respuestas, con el fin de descubrir estrategias y enfoques básicos, de carácter general, para tratar de establecer una organización jerárquica del conocimiento de los expertos.

Análisis

Para aplicar esta técnica se requiere que los expertos resuelvan, en presencia del ingeniero del conocimiento, un conjunto de problemas del dominio. Los problemas planteados deben ser realistas, y el experto comentará todos y cada uno de sus procedimientos de resolución. Las explicaciones deberán ser pormenorizadas, exhaustivas y detalladas. Por su parte, el ingeniero del conocimiento deberá cuestionar cada paso de la resolución efectuada por el experto, y tratará de generalizar las conclusiones y metodologías aplicadas, sin que ello suponga pérdida de la calidad inferencial.

Refinamiento

Es un proceso de análisis invertido, en el que el experto plantea problemas al ingeniero del conocimiento para que éste los resuelva. Los problemas planteados deben ser de dificultad creciente, y el experto debe supervisar al ingeniero del conocimiento mientras trata de resolver los problemas. Si ya existe un prototipo operativo, el proceso de refinamiento se repite con el sistema. El objetivo final es la crítica de planteamientos, la búsqueda de soluciones alternativas y la optimización de los procesos de resolución.

Examen

Supone la revisión microscópica del conocimiento del sistema. Aquí es el experto quien debe analizar todas y cada una de las reglas del sistema, supervisar las estrategias de resolución y dar el "visto bueno" al prototipo construido. Como resultado colateral, el resultado de un buen examen suele ser el incremento y mejora de las posibilidades de explicación y justificación del SE.

Validación

Más que una técnica de extracción es un proceso completo mediante el cual se pretende comprobar que el SE, considerado globalmente, funciona correctamente frente a problemas reales y en entornos reales.

Todas las técnicas de extracción del conocimiento que se acaban de exponer se apoyan en una serie de *herramientas de ayuda*, como son:

Entrevistas Permiten generar conocimientos sobre la terminología del dominio y sobre el universo de discurso que se pretende modelizar, y pueden ser de dos tipos: ***estructuradas*** (con guión) o ***no estructuradas*** (informales).

Análisis de tareas y protocolos Trata de determinar cuáles son las limitaciones impuestas por la naturaleza del problema, cuál es el conocimiento relevante y cuál es la estructura genérica de los diversos problemas del dominio. Para lograr los objetivos propuestos hay que investigar el comportamiento del experto en el marco de las tareas identificadas, analizarlo minuciosamente y conseguir un conjunto de reglas relevantes.

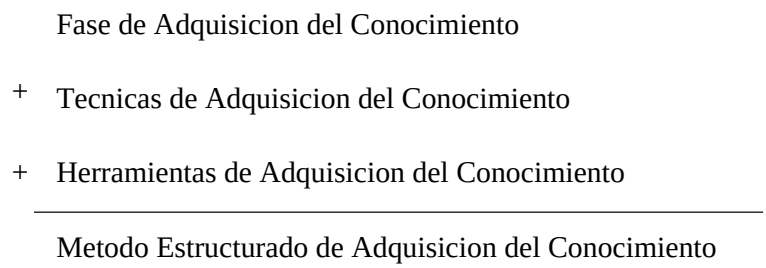
Clasificación de conceptos Es útil porque en todo proceso inteligente, además del conocimiento detallado disponible estrictamente relativo al dominio, los expertos utilizan un conocimiento más global, de muy alto nivel y muy estructurado, que les permite organizar procesos inferenciales complicados. Como consecuencia, tienden a agrupar la información en clases y establecer jerarquías entre ellas.

11.6. Método Estructurado de Adquisición del Conocimiento

La discusión efectuada hasta el momento nos va a permitir proponer no sólo una metodología para la adquisición del conocimiento, sino una metodología completa de IC (ya que el resultado final debería ser un SE perfectamente operativo), en la que podemos distinguir las siguientes fases:

- **Fase Inicial**
  - Realización de entrevistas no estructuradas ni dirigidas
  - Obtención de ejemplos para su análisis conjunto por parte de expertos humanos y de ingenieros del conocimiento
  - Establecimiento de un conjunto inicial de reglas
- **Fase Metodológica**
  - Estructuración macroscópica del conocimiento obtenido en la fase anterior
  - Organización global tentativa del sistema en desarrollo
  - Clasificación de todos los elementos de información que compartan características
- **Fase Estructurada**
  - Organización y estructuración microscópica de los conjuntos de información que comparten características

  - Diseño y construcción de prototipos y módulos individuales
  - Evaluación y refinamiento, por separado, de todos y cada uno de los prototipos y módulos construidos
- **Fase de Evaluación**
  - Integración de módulos
  - Optimización de las estructuras de control
  - Adecuación de las interfaces y de los mecanismos de explicación y de justificación
  - Validación del sistema en el laboratorio
  - Validación del sistema en su entorno real de trabajo
  - Vuelta atrás[10]



[10]Cuando no sea necesaria, el SE será completamente operativo.

Capítulo 12

Verificación y Validación de Sistemas Inteligentes

La **verificación** y **validación** son dos de las etapas más importantes en el análisis del comportamiento de un sistema inteligente. Sin entrar en grandes profundidades, veremos que con la **verificación** trataremos de comprobar si hemos construido nuestro sistema correctamente (es decir, que el *software* implementado no tiene errores y que el producto final satisface los requisitos y las especificaciones de diseño) y con la **validación** nos referiremos a un análisis de la calidad del sistema inteligente en su entorno real de trabajo (lo que nos permitirá determinar si el producto desarrollado satisface convenientemente las expectativas inicialmente depositadas[1]).

Ambas fases forman la base de un entramado más complejo destinado a evaluar globalmente el comportamiento de un sistema inteligente. Por simplicidad, las fases posteriores a la verificación y validación se agrupan bajo el término **evaluación**, que se encarga de analizar aspectos que van más allá de la corrección de las soluciones finales del sistema (utilidad, robustez, velocidad, eficiencia, posibilidades de ampliación, facilidad de manejo, análisis coste vs. beneficio, etc.).

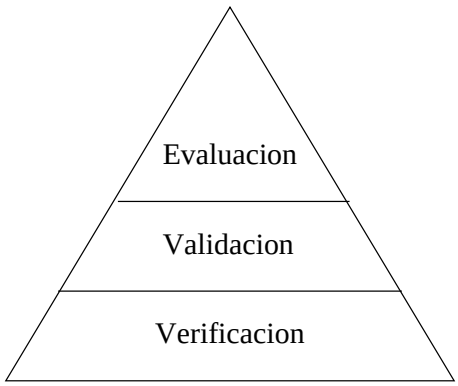


Figura 12.1: Pirámide del análisis del comportamiento de un S.I.

[1] ¿Por qué hay que comprobar si un SE se comporta como un humano? Porque es un modelo computacional del experto del dominio.

12.1. Verificación de Sistemas Inteligentes

La **verificación** de SI[2] es un proceso que incluye las siguientes tareas:

- Verificación del cumplimiento de las especificaciones
- Verificación de los mecanismos de inferencia
- Verificación de la base de conocimientos

12.1.1. Verificación de Especificaciones

El análisis del cumplimiento de las especificaciones puede ser llevado a cabo por los desarrolladores, los usuarios, los expertos y/o un grupo de evaluadores independientes. En el software convencional este proceso está cada vez más automatizado con el advenimiento de las herramientas de ingeniería del software asistida por ordenador (*CASE*), aunque su inclusión en el ámbito de la IC es lenta.

Las cuestiones a analizar en este proceso consisten en comprobar si:

- √ Se ha implementado el paradigma de representación del conocimiento adecuado
- √ Se ha empleado la técnica de razonamiento adecuada
- √ El diseño y la implementación han sido llevados a cabo modularmente
- √ La conexión con el software externo se realiza de forma adecuada
- √ La interfaz de usuario cumple las especificaciones
- √ Las facilidades de explicación son apropiadas para los potenciales usuarios del sistema
- √ Se cumplen los requisitos de rendimiento en tiempo real
- √ El mantenimiento del sistema es posible hasta el grado especificado
- √ El sistema cumple las especificaciones de seguridad
- √ La base de conocimientos está protegida ante modificaciones realizadas por el personal no autorizado

12.1.2. Verificación de Mecanismos de Inferencia

El uso de `shell`s comerciales ha reducido la dificultad de la verificación de los mecanismos de inferencia, ya que se asume que ésta ha sido realizada por los desarrolladores de la herramienta[3]. La responsabilidad del ingeniero del conocimiento recae fundamentalmente en la elección de la herramienta apropiada.

[2] En adelante abreviaremos así *Sistemas Inteligentes*.

[3] Sin embargo, esta asunción no siempre es cierta, sobre todo en versiones nuevas de las herramientas, de modo que en aplicaciones que trabajan en dominios críticos —aquéllos en los que no se puede asumir el coste de un error— el correcto funcionamiento debe verificarse a través de distintas pruebas.

En ocasiones, no obstante, los problemas con las `shell`s comerciales pueden estar causados no por errores en su programación, sino por un desconocimiento de su funcionamiento exacto. Por ejemplo, los procedimientos de resolución de conflictos o los mecanismos de herencia pueden hacer difícil el seguimiento del curso exacto de la inferencia, de forma que aunque el conocimiento estático esté verificado, el funcionamiento final del sistema puede no ser el apropiado.

En caso de que decidamos construir nuestros propios mecanismos de inferencia, será preciso realizar su verificación, pudiendo aplicar para ello las técnicas diseñadas en ingeniería del software. Siempre que sea posible, se recomienda la utilización de mecanismos de inferencia certificados cuyo funcionamiento correcto se haya probado.

12.1.3. Verificación de Bases de Conocimientos

La verificación de la base de conocimientos es plena responsabilidad del ingeniero del conocimiento. Se basa en el concepto de **anomalía**. Una **anomalía** es un uso poco común del esquema de representación del conocimiento, que puede ser considerado como un error potencial (existen anomalías que no constituyen errores, y viceversa).

La verificación de la base de conocimientos no nos asegura que las respuestas de nuestro sistema sean correctas, lo que nos asegura es que el sistema ha sido diseñado e implementado de forma correcta. La mayoría de los estudios sobre este tema se refieren a los sistemas basados en reglas, ya que son los más populares.

Aspectos que se suelen examinar a la hora de verificar una base de conocimientos son la *consistencia* y la *completitud* (ver tabla 12.1, página 140).

Influencia de las medidas de incertidumbre

Las reglas para verificar la consistencia y completitud que aparecen en la tabla 12.1 mencionada son válidas siempre y cuando los sistemas no incluyan incertidumbre. En caso de que sí exista dicha incertidumbre la validez de las pruebas queda en entredicho, ya que, como veremos, situaciones normales pueden ser tomadas como errores.

En sistemas que pretenden medir incertidumbres o grados de asociación (utilizando factores de certidumbre, probabilidades bayesianas o cualquier otro método) es importante verificar que estos valores son consistentes, completos, correctos y no redudantes. Esta tarea se realiza, en primer lugar, asegurándonos que cada regla incluye un factor de incertidumbre y que estos factores cumplen los aspectos de la teoría en la que se basan.

La búsqueda de anomalías en las medidas de incertidumbre de un SI es un proceso que no ha recibido mucha atención por parte de los investigadores, quizá debido al limitado número de SE que hacen uso extensivo de dichas medidas. El modo en que el uso de dichas medidas de incertidumbre puede afectar a la realización de los tests de consistencia y completitud puede verse en los siguientes ejemplos:

- *Redundancia*: si antes no afectaba a la salida del sistema, ahora puede causar graves problemas ya que, al contar la misma información dos veces,

Consistencia	*Reglas redundantes*	p(x)∧q(x)→r(x) q(x)∧p(x)→r(x) (aunque ojo, esto no tiene por qué ser redundante; ejemplo, un sistema que trabaje con información temporal)
	Reglas conflictivas	p(x)∧q(x)→r(x) p(x)∧q(x)→ ¬r(x)
	Reglas englobadas en otras	p(x)∧q(x)→r(x) p(x)→r(x) (la primera es más concreta)
	Reglas circulares	p(x)→q(x) q(x)→r(x) r(x)→p(x)
	Condiciones IF innecesarias	p(x)∧q(x)→r(x) p(x)∧ q̸(x)→r(x)
Completitud	*Valores no referenciados de atributos*	Ocurre cuando algunos valores del conjunto de posibles valores de un atributo no son cubiertos por la premisa de ninguna otra regla.
	Valores ilegales de atributos	Una regla referencia valores de atributos que no están incluidos en el conjunto de valores válidos para ese atributo.
	Reglas inalcanzables	p(x)→r(x) p(x) no aparece como conclusión de otra regla ni puede obtenerse del exterior (razonamiento progresivo)
	Reglas sin salida	p(x)∧q(x)→r(x) r(x) no es una conclusión final y no aparece en la premisa de ninguna otra regla (razonamiento proogresivo)

Cuadro 12.1: Verificación de la consistencia y completitud en bases de conocimientos.

se pueden modificar los pesos de las conclusiones.

- *Reglas englobadas en otras*: esta situación puede no ser errónea ya que las dos reglas pueden indicar la misma conclusión pero con distintas confianzas. La regla englobada sería un refinamiento de la regla más general para el caso de que tengamos más información.
- *Reglas circulares*: pueden existir casos en los que la utilización de medidas de incertidumbre rompan la circularidad de un conjunto de reglas. Por ejemplo, si el factor de certidumbre de una conclusión implicada en el ciclo cae por debajo de un umbral (normalmente entre $-0{,}2$ y $0{,}2$) se considera que el valor de la conclusión es "desconocido" y el ciclo se rompe.
- *Condiciones IF innecesarias*: igual que en el caso de las reglas englobadas en otras, una condición IF innecesaria puede utilizarse para variar la confianza en la conclusión final.
- *Reglas inalcanzables*: es un caso muy frecuente, que de forma similar al caso de las reglas sin salida, puede ocurrir que existan reglas que por causa de los factores de certidumbre se conviertan en inalcanzables.
- *Reglas sin salida*: la detección de este tipo de reglas se complica con la introducción de la incertidumbre. Así, una regla puede convertirse en una regla sin salida si su conclusión tiene una certidumbre por debajo del umbral en el cual un valor se considera "conocido".

Verificación dependiente o independiente del dominio

La verificación de un SI puede enfocarse desde dos puntos de vista diferentes: **verificación dependiente del dominio** y **verificación independiente del dominio**. La primera se basa en la detección de las anomalías a través de técnicas heurísticas mediante las cuales se analiza la base de conocimientos pero sin tener en consideración el dominio de aplicación. Por el contrario, la segunda utiliza *metaconocimiento* del propio universo de discurso para examinar la bases de conocimiento implementadas[4]. El inconveniente de este procedimiento es que el metaconocimiento, al no ser más que conocimiento sobre conocimiento, también debe ser verificado. Además, puede no ser estable, si existe aportación continua de nuevo conocimiento, y por último, el desarrollo de una aplicación que permita realizar verificaciones dependientes del dominio suele ser una tarea lenta y costosa, en parte por el hecho de tener que adquirir el metaconocimiento necesario y en parte por tener que mantenerlo.

Automatización de los mecanismos de verificación

De las distintas fases que componen el análisis del comportamiento de un sistema inteligente, la fase de verificación es en la que se ha conseguido un mayor grado de automatización mediante distintos tipos de herramientas. Dentro de estas herramientas de

[4]Un ejemplo de este tipo de verificación es el sistema TEIRESIAS, que supervisa la introducción de conocimiento en el SE MYCIN.

verificación podemos establecer dos grupos: las **dependientes del dominio** (que hacen uso del metaconocimiento) y las **independientes del dominio** (que se basan principalmente en convertir la base de conocimientos en una representación independiente, mediante tablas o grafos, a partir de la que se buscan las posibles anomalías).

12.2. Validación de Sistemas Inteligentes

Una vez verificado el "software" del sistema, el proceso debe continuar con la **validación** del producto. Recordemos que *validar* un SI supone analizar si los resultados del sistema son correctos y por lo tanto se comporta como un experto más en un dominio de aplicación concreto, y si se cumplen las necesidades y los requisitos del usuario.

La **validación** puede verse desde dos óptimas diferentes:

Validación orientada a los resultados Su objetivo es comparar el rendimiento del sistema con un rendimiento esperado (proporcionado por una referencia estándar o por expertos humanos) y comprobar que el sistema alcanza un nivel que se considera aceptable.

Validaciíon orientada al uso Se centra en cuestiones que hacen referencia a la relación hombre-máquina, más allá de la corrección de los resultados obtenidos por el sistema.

Normalmente la *validación orientada a los resultados* es un prerrequisito para la realización de una *validación orientada al uso*. Así, si un sistema no presenta un rendimiento aceptable, los aspectos concernientes a la validación orientada al uso son irrelevantes[5].

12.2.1. Principales características del proceso de Validación

Al estudiar la Validación nos damos cuenta de que no existe una clasificación global de los problemas a resolver ni tampoco existe una clara relación entre estos problemas y las técnicas destinadas a solucionarlos. Entre los principales problemas existentes en la Validación cabe destacar la falta de métricas de evaluación prácticas y rigurosas, la falta de especificaciones, que conduce a evaluaciones subjetivas, y la falta de herramientas adecuadas.

El proceso de Validación presenta distintos problemas para el ingeniero del conocimiento, que debe conocer las distintas aproximaciones para su eventual solución:

Personal involucrado en la validación El primer elemento a considerar es el ingeniero del conocimiento que ha desarrollado el sistema (ya que es quien mejor conoce las características del SI, aunque puede que no sea totalmente objetivo); también es necesario contar con expertos humanos (ya que el método básico para realizar la validación es el análisis de

[5]Por este motivo muchos autores incluyen la validación orientada al uso como una de las primeras fases de la evaluación, refiriéndose con validación sólo a la validación orientada a resultados.

casos de prueba ya resueltos, con los que se estudiarán las discrepancias encontradas), preferiblemente en general ajenos a los que colaboraron en el desarrollo del sistema (para conseguir que el conocimiento del sistema se adecúe al de un consenso de expertos y no únicamente al del experto colaborador, aunque su independencia también puede predisponer a la *falacia del superhombre*[6]); y por último, los usuarios finales, aunque en fases posteriores, cuando el conocimiento ya esté validado.

Partes del sistema a validar Nuestro principal objetivo es lograr que los resultados finales del SI sean correctos, aunque también es interesante analizar si los resultados intermedios son correctos (ya que los finales dependen de ellos) o si el razonamiento seguido hasta dar con la solución (sus estructuras) es el apropiado (ya que un proceso de razonamiento incorrecto puede provocar errores cuando queramos ampliar nuestra base de conocimientos).

Datos utilizados en la validación El uso de casos de prueba es el método más ampliamente utilizado para la validación de SE. En un mundo ideal contaríamos con una gran cantidad de casos que representarían un rango completo de problemas que serían analizados por una serie de expertos, pero en la realidad desafortunadamente es muy común no disponer más que de un número reducido de casos y con pocos expertos que nos ayuden a analizarlos. Para que una muestra de casos sea susceptible de ser aceptada en un proceso de validación debe cumplir dos propiedades fundamentales: *cantidad* (para que las medidas de rendimiento que obtengamos sean estadísticament significativas) y *representatividad* (no sólo hay que capturar un número elevado de casos, sino que deben ser representativos de los problemas comunes a los que se va a enfrentar el SI). La cobertura de casos es mucho más importante que su número, y deben representar con fiabilidad el dominio de entrada del sistema (casos susceptibles de ser tratados). En el caso extremo de no disponer de casos de prueba para validar el sistema (ya que no es aconsejable utilizar los casos empleados en el diseño) se pueden usar *casos sintéticos*, generados artificialmente por los expertos.

Criterios de validación Podemos diferenciar dos tipos de validación atendiendo al tipo de criterio establecido: *validación contra el experto* (se utilizan las opiniones y diagnósticos de expertos humanos como criterio de validación; inconvenientes: subjetividad) y *validación contra el problema* (contrastar los resultados del sistema con la situación real; inconvenientes: falacia del superhombre, no disponibilidad de la solución real).

Momento en que se realiza la validación El punto de vista más comúnmente aceptado es el de realizar la validación a lo largo del desarrollo del sistema, realizando preferentemente un desarrollo incremental en el

[6]Exigir más al SI de lo que se exigiría a un experto humano.

cual, al final de cada incremento, se realiza una validación. La validación que se realiza en etapas tempranas del desarrollo está muy vinculada al proceso de adquisición del conocimiento (*refinamiento del conocimiento*).

Otro aspecto a tener en cuenta consiste en la diferenciación entre:

Validación retrospectiva Se realiza sobre casos históricos ya resueltos y almacenados en una base de datos. Es el tipo de validación más comúnmente realizada en los SE, pudiendo ser una validación contra expertos o contra el problema. Se utiliza en las etapas de desarrollo del sistema, antes de que éste se instale en su campo de trabajo habitual.

Validación prospectiva Consiste en confrontar al sistema con casos reales y ver si es capaz de resolverlos o no. No se utilizan casos almacenados en una base de datos, sino casos que en este momento están siendo tratados por expertos humanos (se relaciona con la validación contra el problema). El inconveniente surge, asimismo, cuando el dominio de aplicación es crítico y el coste de una decisión errónea no es asumible. Suele utilizarse, no obstante, cuando ya se ha validado retrospectivamente el sistema y se desea realizar una nueva validación en el campo de aplicación.

Métodos de validación Los métodos para realizar la validación se pueden dividir en dos grupos principales:

Métodos cualitativos Emplean técnicas subjetivas de comparación de rendimiento: validación de superficie (proceso informal de discusión y análisis entre expertos e ingenieros del conocimiento), prueba de Turing, test de campo (exposición del sistema a los usuarios), validación de subsistemas, análisis de sensibilidad (presentación de entradas muy similares).

Métodos cuantitativos Se basan en medidas estadísticas:

- *Medidas de pares* (métodos de validación contra expertos). Se dividen en medidas de acuerdo y medidas de asociación.
  Medidas de acuerdo:
  - Índice de acuerdo (cociente entre el número de observaciones de acuerdo y las totales):

$$I = \frac{\sum_{i=j}^{k} n_{ij}}{N} = \sum_{i=j}^{k} p_{ij}$$

Ventaja: interpretación sencilla. Inconveniente: no diferencia los desacuerdos según su importancia y no tiene en cuenta la casualidad.

- Índice de acuerdo contra uno (similar al anterior, considera acuerdos parciales, los que se diferencian en una sola categoría):

$$I = \frac{\sum_{\substack{i=j \\ i=j\pm 1}}^{k} n_{ij}}{N} = \sum_{\substack{i=j \\ i=j\pm 1}}^{k} p_{ij}$$

  Ventaja: elimina problemas asociados a las categorías semánticas ordinales con límites poco claros y permite en análisis de tendencias optimistas o pesimistas (respectivamente, por encima y por debajo de la diagonal de acuerdo).

- Kappa (corrige acuerdos debidos a la casualidad):

$$k = \frac{p_0 - p_c}{1 - p_c}$$

  donde p_0 es la proporción de acuerdo observado y p_c la proporción de acuerdo esperado debido a la casualidad, de modo que $1 - p_c$ es el máximo acuerdo posible una vez eliminada la casualidad y $p_0 - p_c$ es el acuerdo obtenido una vez eliminada la casualidad.
  Esto es, si $k < 0$ el índice de acuerdo es menor aún que el esperado debido a la casualidad, si $k = 0$ el índice de acuerdo es el esperado debido a la casualidad y si $k = 1$ el acuerdo es completo, independientemente de la casualidad.

$$p_c = \sum_{i=j}^{k} p_{i.} p_{.j}$$

  Problema: trata todos los desacuerdos de la misma forma.

- Kappa ponderada (corrige el problema de la Kappa):

$$k_w = 1 - \frac{\sum_{i=1,j=1}^{k} v_{ij} p_{oij}}{\sum_{i=1,j=1}^{k} v_{ij} p_{cij}}$$

  Medidas de asociación miden el grado de asociación lineal entre el sistema y el experto humano (tau de Kendall, rho de Spearman,...).

- *Medidas de grupo.* Las medidas de pares son útiles cuando el número de expertos es reducido (para cada par de expertos hay

que hacer una tabla de contingencia), pero si la validación involucra un grupo amplio de expertos, la información que proporcionan las medidas de pares puede resultar difícil de interpretar. En todo caso, las medidas de pares sirven de base para este otro tipo de medidas, cuyo objetivo es analizar conjuntamente las interpretaciones de los expertos y tratar de buscar estructuras de representación que permitan una interpretación más sencilla dentro del contexto de la validación.
El procedimiento para obtener medidas de grupo es obtener medidas de pares para cada uno de los posibles pares de expertos de la validación, agrupar los resultados en una tabla resumen y obtener la medida de grupo a partir de éstas. Algunas medidas de grupo son:

- Índice de Williams:

$$I_0 = \frac{P_0}{P_n}$$

donde P_0 representa el acuerdo existente entre un experto aislado en relación a un grupo de expertos de referencia y P_n representa el acuerdo existente dentro de dicho grupo de referencia. Se definen a su vez:

$$P_0 = \frac{\sum_{a=1}^{n} P_{(0,a)}}{n}$$

$$P_n = \frac{\sum_{a=1}^{n-1} \sum_{b=a+1}^{n} P_{(a,b)}}{n(n-1)}$$

La interpretación de I_n es que si $I_n < 1$ el acuerdo entre el experto aislado y el grupo de expertos es menor que el acuerdo entre los propios miembros del grupo, si $I_n = 1$, el experto aislado coincide con el grupo al mismo nivel que los miembros del grupo entre sí y si $I_n > 1$ el experto aislado coincide con el consenso del grupo.

- Análisis cluster. Su objetivo es establecer grupos de expertos según su grado de concordancia e identificar a cuál se parece más nuestro SI, existiendo para ello dos variantes: métodos jerárquicos y métodos no jerárquicos.
La aplicación de un método jerárquico de análisis cluster implica la construcción de una matriz de concordancia que describa las distancias entre todos los miembros involucrados en el estudio. Una distancia apropiada podría ser, por ejemplo, los índices de acuerdo encontrados entre los distintos expertos. A partir de los datos de la matriz de concordancia podemos establecer una secuencia de agrupamientos anidados que definen una estructura en árbol denominada *dendrograma*, en la que cada nivel representa una partición del conjunto global de los elementos que son objeto del análisis.

Por su parte, los métodos no jerárquicos de análisis cluster realizan una clasificación en la que se minimiza la suma de los cuadrados de las distancias entre cada punto y el centroide de su clase. Para ello hay que predefinir un número arbitrario de clústeres, situar aleatoriamente los centroides de cada uno, asignar cada punto al centroide más cercano y reevaluar iterativamente las posiciones de los nuevos centroides de cada clúster. La mayor dificultad es la interpretación del concepto de coordenadas de puntos, por lo que generalmente se prefiere el uso de métodos jerárquicos.

- Ratios de acuerdo. Miden el acuerdo existente entre un experto (o SI) y una referencia estándar (que puede ser un consenso entre expertos —validación contra el experto— o la solución real al problema planteado —validación contra el problema—). Problema: no siempre existe o puede definirse una referencia estándar (sólo si existe un sistema físico que evoluciona).

Errores en la validación En el proceso de validación se pueden dar dos tipos de errores: *errores de Tipo I* (cuando el sistema es considerado como no válido aun a pesar de serlo, se denominan también de "riesgo para el desarrollador") y *errores de Tipo II* (cuando se acepta como válido un sistema que no lo es; siendo más peligrosos que el caso anterior, se denominan también de "riesgo para el usuario").

		Referencia Estándar		
		D	$\neg D$	
Experto	D	a	b	$a+b$
	$\neg D$	c	d	$c+d$
		$a+c$	$b+d$	

Cuadro 12.2: Tabla de contingencia

	Ratios de Acuerdo	
	Ratio de verdaderos positivos	$\frac{a}{a+c}$
	Ratio de verdaderos negativos	$\frac{d}{b+d}$
Sensibilidad	Ratio de falsos positivos	$\frac{b}{b+d}$
	Ratio de falsos negativos	$\frac{c}{a+c}$
Especifidad	Valor predictivo positivo	$\frac{a}{a+b}$
	Valor predictivo negativo	$\frac{d}{c+d}$

Cuadro 12.3: Ratios de Acuerdo

Otras medidas de Similitud	
Índice de acuerdo	$\frac{a+d}{a+b+c+d}$
Coeficiente de Jaccard	$\frac{a}{a+b+c}$

Cuadro 12.4: Medidas de Similitud

12.2.2. Metodología de Validación

Como hemos visto, la **validación** no es un proceso sencillo de aplicar. Para facilitar su ejecución, puede dividirse en tres fases claramente diferenciadas:

Planificación Fase de análisis de características del dominio de aplicación, del sistema y de la etapa de desarrollo en la que se encuentre el sistema para establecer una serie de estrategias de validación.

Aplicación Fase en la que se llevan a la práctica las estrategias establecidas en la fase anterior y se aplican medidas cuantitativas que puedan darse dentro de un contexto cualitativo.

Es necesario realizar una captura de casuística de validación suficiente y representativa. Generalmente los casos de prueba deberán ser preprocesados para corregir errores, transformar datos a representaciones más adecuadas e incluir información adicional (formato de la BB.DD., orden de las categorías semánticas, pesos de desacuerdo,...). Una vez realizadas la captura y el preprocesado de la casuística se realizan las medidas cuantitativas.

Interpretación Fase en la que se utilizan los resultado de la anterior para dilucidar si el SI se comporta realmente como un experto dentro de su campo de aplicación. Es la fase más compleja de la metodología porque los resultados de los tests estadísticos deben tener en cuenta la naturaleza del problema que estamos tratando y las características de la muestra empleada en su obtención.

Índice alfabético

Índice de figuras

Índice de cuadros

Bibliografía

[1] Moret Bonillo, Vicente y Alonso Betanzos, Amparo y Cabrero Canosa, Mariano y Guijarro Berdiñas, Bertha y Mosqueira Rey, Eduardo.
Fundamentos de Inteligencia Artificial.
Servicio de Publicaciones de la Universidad de La Coruña, Octubre 2000.
2^a edición.