

Facsímil 12

Lógica e incertidumbre

Coleccionable completo · 7 capítulos

Índice

01	Lógica para máquinas: proposicional, primer orden y resolución
02	Razonamiento bajo incertidumbre: Bayes y redes bayesianas
03	Cuando la verdad es difusa: factores de certeza y lógica difusa
04	Sistemas expertos y soporte a la decisión
05	Razonamiento causal: de la correlación a la intervención
06	Razonamiento no monótono y revisión de creencias
07	Lo que deberías saber: lógica, conocimiento e incertidumbre

Capítulo 01: Lógica para máquinas: proposicional, primer orden y resolución

Entrando en el tema

Discutes con alguien sobre el horario de un vuelo. Tú dices: «Si el vuelo sale tarde, perdemos la conexión. Y el vuelo sale tarde». La otra persona, sin mirar nada más, concluye: «Entonces perdemos la conexión». Nadie ha consultado la pantalla de salidas. La conclusión no viene de los datos, viene de la forma del razonamiento. Da igual que hablemos de vuelos, facturas o reactores nucleares: si aceptas las dos primeras frases, la tercera es inevitable.

Esa inevitabilidad es lo que estudia la lógica. No le importa de qué hablas, le importa cómo encajan las afirmaciones. Y esa indiferencia al contenido es justo lo que la hace mecanizable: si una conclusión se sigue por pura forma, una máquina puede comprobarlo sin entender de qué va el asunto.

Este capítulo trata de cómo se le enseña a razonar a una máquina con garantías. No con plausibilidad, no con «suena bien», sino con la clase de certeza que tiene « $2 + 2 = 4$ ». Es la base de la IA simbólica, la rama que durante décadas fue *la* inteligencia artificial, y que hoy sigue sosteniendo cosas muy concretas: los solvers SAT del Facsímil 2, los validadores que aprueban o rechazan la salida de un modelo, las ontologías OWL que dicen qué se puede inferir de un grafo. Cuando un agente del Facsímil 5 decide que una acción está permitida, por debajo hay reglas que se evalúan con esta maquinaria.

Vamos a recorrer tres niveles. Primero la lógica proposicional: enunciados enteros que son verdaderos o falsos. Después la regla de resolución, el único mecanismo de inferencia que necesita una máquina para demostrar cualquier cosa demostrable. Y por último la lógica de primer orden, que añade objetos, relaciones y cuantificadores, y nos lleva directos a Prolog y a la programación lógica.

Lógica proposicional: enunciados que valen verdadero o falso

La pieza más pequeña de la lógica proposicional es la **proposición**: una afirmación que es verdadera o falsa, sin medias tintas. «Llueve», «el usuario es administrador», «el importe supera el límite». No nos interesa su contenido interno, solo su valor de verdad. Por eso usamos letras: p , q , r , o nombres más legibles como *admin*, *borrar*.

A partir de ahí, combinamos proposiciones con **conectivas**. Son cinco, y conviene conocerlas con precisión porque toda la maquinaria descansa en ellas.¹

CONECTIVA	SÍMBOLO	LECTURA	EJEMPLO COTIDIANO
Negación	$\neg$	«no»	$\neg admin$: el usuario no es administrador.
Conjunción	$\wedge$	«y»	$admin \wedge activo$: es administrador y está activo.
Disyunción	$\vee$	«o» (inclusivo)	$email \vee banner$: sale por email, por banner o por ambos.
Implicación	$\rightarrow$	«si... entonces»	$admin \rightarrow borrar$: si es administrador, puede borrar.
Bicondicional	$\leftrightarrow$	«si y solo si»	$activo \leftrightarrow pagado$: activo exactamente cuando ha pagado.

La **sintaxis** dice qué cadenas están bien formadas: una variable es una fórmula; si φ es fórmula, $\neg\varphi$ también lo es; si φ y ψ lo son, también lo son $(\varphi \wedge \psi)$, $(\varphi \vee \psi)$, $(\varphi \rightarrow \psi)$ y $(\varphi \leftrightarrow \psi)$. Nada más es una fórmula. Esta definición recursiva es la misma idea que un árbol de sintaxis en un compilador: las fórmulas se construyen unas dentro de otras.

La **semántica** dice qué significan. Una **interpretación** (o asignación) es una función que da verdadero o falso a cada variable:

$$v : \{p_1, p_2, \dots, p_n\} \rightarrow \{V, F\}$$

En palabras: fijar una interpretación es rellenar, una por una, las casillas de «verdadero» o «falso» de cada variable; es elegir uno de los mundos posibles entre los que la fórmula podría vivir.

SÍMBOLO	SIGNIFICADO	EJEMPLO
v	Interpretación: una forma concreta de fijar los valores.	$v(admin) = V$, $v(activo) = F$.
p_i	Variable proposicional.	$admin$, $borrar$.
$\{V, F\}$	Los dos valores de verdad posibles.	Verdadero o falso.

Dada una interpretación, el valor de cualquier fórmula se calcula de dentro hacia fuera siguiendo las tablas de cada conectiva. La conectiva que más sorprende al principio es la implicación: $admin \rightarrow borrar$ solo es **falsa** cuando $admin$ es verdadero y $borrar$ es falso. Si el usuario no es administrador, la promesa «si eres administrador, puedes borrar» no se ha roto: simplemente no aplica, y por convención la consideramos verdadera. Esa convención (la implicación material) es la que permite razonar con reglas sin tener que comprobar antes si se dan sus condiciones.

Tablas de verdad, tautologías y la diferencia entre validez y satisfacibilidad

Una **tabla de verdad** enumera todas las combinaciones posibles de valores de las variables y, para cada una, da el valor de la fórmula. Con n variables hay 2^n filas. Esta es la tabla con las cinco conectivas a la vez:

A	B	$\neg A$	$A \wedge B$	$A \vee B$	$A \rightarrow B$	$A \leftrightarrow B$
V	V	F	V	V	V	V
V	F	F	F	V	F	F
F	V	V	F	V	V	F
F	F	V	F	F	V	V

La tabla de verdad es el método más bruto y más seguro que existe: si quieres saber qué hace una fórmula, la calculas en sus cuatro, ocho o dieciséis filas. El problema es que crece de forma explosiva. Con 30 variables ya tienes más de mil millones de filas. Esa explosión es exactamente la frontera de complejidad de SAT que vimos en el Facsímil 2: decidir si una fórmula proposicional tiene algún modelo es el primer problema que se demostró NP-completo.²

Con la tabla en la mano podemos clasificar cualquier fórmula:

CLASE	DEFINICIÓN	EJEMPLO
Tautología (válida)	Verdadera en todas las interpretaciones.	$A \vee \neg A$: o llueve o no llueve.
Contradicción (insatisfacible)	Falsa en todas las interpretaciones.	$A \wedge \neg A$: llueve y no llueve.
Satisfacible	Verdadera en al menos una interpretación.	$A \wedge B$: basta con $A = V, B = V$.
Contingente	Verdadera en unas y falsa en otras.	$A \rightarrow B$: depende de A y B .

Aquí aparece una de las distinciones más importantes del capítulo, y la que más confusión genera:

- **Validez** pregunta: ¿es verdadera *siempre*? Es una propiedad fuerte. Una fórmula válida no aporta información sobre el mundo, porque se cumple pase lo que pase.

- **Satisfacibilidad** pregunta: ¿es verdadera *alguna vez*? Es una propiedad débil. Que algo sea satisfacible solo dice que no es contradictorio.

Las dos están conectadas por la negación, y esta equivalencia es el puente que usan los demostradores automáticos:

$$\varphi \text{ es válida} \iff \neg\varphi \text{ es insatisfacible}$$

En palabras: demostrar que algo se cumple pase lo que pase es exactamente lo mismo que demostrar que su contrario no puede cumplirse en ningún escenario.

Léelo despacio, porque es la idea sobre la que se construye todo lo que viene después. Si quiero demostrar que algo es verdadero siempre, me basta con demostrar que su negación no puede ser verdadera nunca. En lugar de comprobar infinitas (o exponencialmente muchas) interpretaciones buenas, busco una sola contradicción. Un solver SAT, que solo sabe responder «satisfacible» o «insatisfacible», se convierte así en un demostrador de teoremas: para probar φ , le pides que intente satisfacer $\neg\varphi$; si fracasa, φ queda demostrada.

Pruébalo aquí mismo: calculadora de tablas de verdad

No hace falta esperar al cuaderno para ver esto funcionar. Escribe una fórmula y la tabla aparece al instante, con su clasificación: **tautología** (verdadera en todas las filas), **contradicción** (falsa en todas) o **contingente** (depende de los valores). Prueba con $p \vee \neg p$ (tautología), $p \wedge \neg p$ (contradicción) o el clásico $(p \rightarrow q) \wedge (q \rightarrow r)$.

Cambia la fórmula y observa: cuando algo es una tautología, ninguna fila de la última columna es falsa; cuando lo niegas, ninguna es verdadera. Esa es, en miniatura, la maquinaria que un *solver* SAT hace a gran escala sin enumerar todas las filas.

Equivalencias: reescribir sin cambiar el significado

Dos fórmulas son **equivalentes** ($\equiv$) si tienen la misma tabla de verdad. Equivalente no significa idéntico: significa intercambiable, como $\frac{2}{4}$ y $\frac{1}{2}$. Estas son las equivalencias que más se usan al preparar fórmulas para una máquina:

NOMBRE	EQUIVALENCIA	PARA QUÉ SIRVE
Eliminar implicación	$A \rightarrow B \equiv \neg A \vee B$	Quitar el $\rightarrow$, que un solver no consume directamente.
Contrapositiva	$A \rightarrow B \equiv \neg B \rightarrow \neg A$	Razonar hacia atrás (es la base de modus tollens).
Doble negación	$\neg\neg A \equiv A$	Simplificar.
De Morgan	$\neg(A \wedge B) \equiv \neg A \vee \neg B$	Empujar la negación hacia dentro.
De Morgan	$\neg(A \vee B) \equiv \neg A \wedge \neg B$	Lo mismo, dual.
Distributiva	$A \vee (B \wedge C) \equiv (A \vee B) \wedge (A \vee C)$	Convertir a forma normal conjuntiva.

Forma normal conjuntiva: el formato que las máquinas entienden

Una máquina no quiere fórmulas con paréntesis anidados y conectivas mezcladas. Quiere un formato plano y uniforme. Ese formato es la **forma normal conjuntiva** (CNF, por sus siglas en inglés): una conjunción de cláusulas, donde cada cláusula es una disyunción de literales, y un literal es una variable o su negación.

$$\varphi = \bigwedge_{j=1}^m C_j, \quad C_j = \bigvee_{\ell \in L_j} \ell, \quad \ell \in \{p_i, \neg p_i\}$$

SÍMBOL O	SIGNIFICADO	EJEMPLO
$\bigwedge_{j=1}^m$	Conjunción de las m cláusulas: todas deben cumplirse a la vez.	Las dos cláusulas de la regla de permisos juntas.
C_j	La cláusula número j , una disyunción de literales.	$(\neg admin \vee borrar)$.
$\bigvee_{\ell \in L_j}$	Disyunción dentro de la cláusula: basta con un literal verdadero.	<i>borrar</i> o $\neg admin$.
ℓ	Un literal: una variable o su negación.	<i>admin</i> , $\neg admin$.

En palabras: una fórmula en forma normal conjuntiva es una lista de exigencias unidas por «y», donde cada exigencia es un menú de opciones unidas por «o»; se cumple solo si en cada cláusula al menos un literal sale verdadero.

Es exactamente la misma CNF del capítulo de SAT del Facsímil 2: una lista de cláusulas que deben cumplirse todas, donde cada cláusula se satisface si al menos uno de sus literales es verdadero. Toda fórmula proposicional se puede llevar a CNF con tres pasos mecánicos:

1. Eliminar implicaciones y bicondicionales con las equivalencias de arriba.
2. Empujar las negaciones hasta pegarlas a las variables, usando De Morgan y doble negación.
3. Distribuir $\vee$ sobre $\wedge$ hasta que la fórmula sea una conjunción de disyunciones.

Veámoslo con una regla de permisos. Queremos pasar a CNF la frase «si Ana es administradora, entonces puede borrar y queda registrada en auditoría»:

$$admin \rightarrow (borrar \wedge auditar)$$

PASO	RESULTADO	QUÉ HEMOS HECHO
Partida	$admin \rightarrow (borrar \wedge auditar)$	La regla original.
Quitar $\rightarrow$	$\neg admin \vee (borrar \wedge auditar)$	Aplicamos $A \rightarrow B \equiv \neg A \vee B$.
Distribuir	$(\neg admin \vee borrar) \wedge (\neg admin \vee auditar)$	Repartimos el $\vee$ sobre el $\wedge$.

El resultado son dos cláusulas: $(\neg admin \vee borrar)$ y $(\neg admin \vee auditar)$. Una sola regla de negocio se ha convertido en dos cláusulas limpias, listas para un solver o para la regla de resolución que viene ahora.

Inferencia: de las reglas a la resolución

Hasta aquí hemos descrito fórmulas. Pero el objetivo de la lógica es **inferir**: a partir de unas premisas que aceptamos, obtener conclusiones que estábamos obligados a aceptar. La relación clave se llama **consecuencia lógica** y se escribe con la doble barra:

$$\Gamma \models \psi$$

SÍMBOLO	SIGNIFICADO	EJEMPLO
Γ	El conjunto de premisas que aceptamos como verdaderas.	$\{admin, admin \rightarrow borrar\}$.
$\models$	«tiene como consecuencia lógica»: la doble barra.	Lo que separa premisas de conclusión.
ψ	La conclusión que queremos justificar.	$borrar$.

En palabras: siempre que todas las premisas sean verdaderas, la conclusión también lo es; no existe ningún escenario que cumpla las premisas y falle la conclusión.

Significa: en toda interpretación que hace verdaderas todas las fórmulas del conjunto Γ , también ψ es verdadera. No dice « ψ es verdadera», dice « ψ es verdadera siempre que lo sean las premisas». Es el «entonces perdemos la conexión» del principio del capítulo: no afirmamos nada sobre el mundo, afirmamos que la conclusión va incluida en lo que ya hemos aceptado.

Reglas clásicas: modus ponens y modus tollens

Durante siglos, razonar fue aplicar reglas con nombre. Dos de ellas siguen siendo el pan de cada día:

REGLA	FORMA	LECTURA
Modus ponens	De A y $A \rightarrow B$, deriva B .	«Es administrador; si es administrador puede borrar; luego puede borrar.»
Modus tollens	De $\neg B$ y $A \rightarrow B$, deriva $\neg A$.	«No quedó registrado en auditoría; si fuera administrador habría quedado; luego no es administrador.»

Modus ponens razona hacia delante; modus tollens, hacia atrás por la contrapositiva. Son válidas y útiles, pero tienen un problema para una máquina: hay decenas de reglas de este estilo, y elegir cuál aplicar en cada momento es un arte. Sería mucho mejor disponer de **una sola regla** que, aplicada mecánicamente, baste para demostrar cualquier consecuencia. Esa regla existe.

La regla de resolución

En 1965, John Alan Robinson publicó una regla de inferencia pensada para máquinas, no para humanos.³ La idea es de una simplicidad desconcertante. Si en una cláusula aparece un literal ℓ y en otra aparece su complementario $\neg\ell$, esos dos no pueden ser verdaderos a la vez, así que se cancelan, y el resto de ambas cláusulas se une en una cláusula nueva:

$$\frac{(\ell \vee A) \quad (\neg\ell \vee B)}{(A \vee B)}$$

SÍMBOLO	SIGNIFICADO
ℓ	Literal que aparece afirmado en una cláusula.
$\neg\ell$	Su complementario, que aparece en la otra.
A, B	El resto de literales de cada cláusula (pueden estar vacíos).
$(A \vee B)$	El resolvente : la cláusula nueva que se deriva.

En palabras: si una cláusula afirma algo y otra lo niega, ese algo no puede inclinar la balanza, así que se tacha y los literales que quedan en ambas se juntan en una cláusula nueva.

Un caso particular merece nombre propio. Cuando una de las cláusulas es un solo literal, la regla se llama **resolución unitaria**, y es justo la propagación unitaria de DPLL del Facsímil 2: de (ℓ) y $(\neg\ell \vee B)$ se obtiene (B) . Modus ponens, de hecho, no es más que un caso de resolución: A es la cláusula (A) , $A \rightarrow B$ es la cláusula $(\neg A \vee B)$, y al resolverlas sale (B) . Una sola regla absorbe a las clásicas.

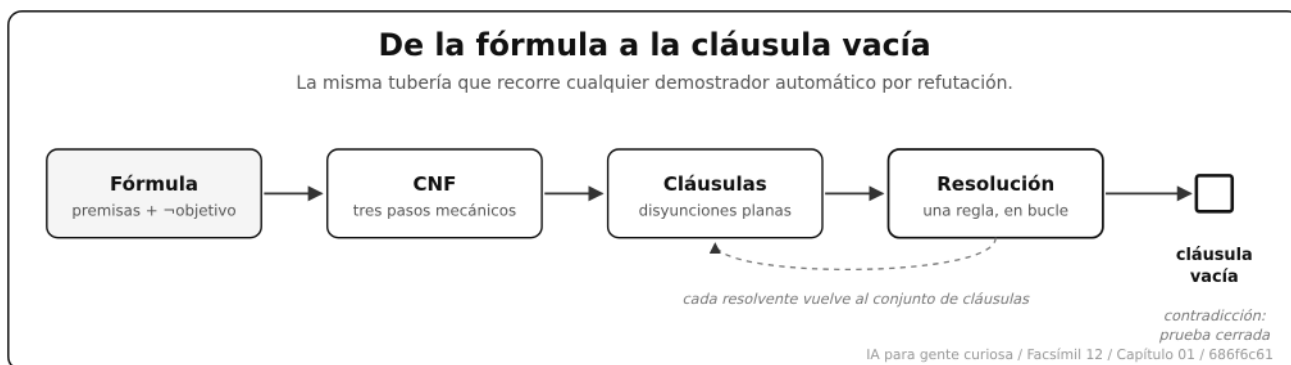
¿Y qué pasa cuando resolvemos (ℓ) con $(\neg\ell)$, dos literales solos y opuestos? No queda nada. El resolvente es la **cláusula vacía**, que se escribe como un cuadro vacío. Una cláusula es verdadera si alguno de sus literales lo es; la cláusula vacía no tiene literales, así que no puede ser verdadera nunca. Es la contradicción hecha símbolo. Encontrarla es la señal de que algo en nuestro conjunto de cláusulas es imposible.

Demostrar por refutación

Esto enlaza con el puente que vimos antes (validez es insatisfacibilidad de la negación) y da el método estrella de la lógica computacional: la **prueba por refutación**. Para demostrar que ψ se sigue de unas premisas Γ , no intentamos derivar ψ directamente. Hacemos lo contrario:⁴

1. Suponemos lo contrario de lo que queremos probar: añadimos $\neg\psi$ a las premisas.
2. Pasamos todo a CNF: $\Gamma \wedge \neg\psi$ se convierte en un conjunto de cláusulas.
3. Aplicamos resolución una y otra vez, derivando resolventes.
4. Si llegamos a la cláusula vacía, hemos encontrado una contradicción. Eso significa que $\Gamma \wedge \neg\psi$ es insatisfacible, y por tanto $\Gamma \models \psi$.

Visto de lejos, todo el método es una sola tubería: una fórmula entra por un extremo y, si la conclusión se seguía de verdad, sale por el otro convertida en la cláusula vacía.



Suena a truco, pero es la forma de razonar de todo demostrador automático serio. Tomemos un ejemplo concreto, del mundo de los permisos que tanto aparece en el facsímil. Las premisas:

- «Si Ana es administradora, puede borrar»: $admin \rightarrow borrar$, es decir la cláusula $(\neg admin \vee borrar)$.
- «Si puede borrar, la acción queda registrada en auditoría»: $borrar \rightarrow auditar$, es decir $(\neg borrar \vee auditar)$.
- «Ana es administradora»: $(admin)$.

Queremos demostrar: *auditar*, o sea, que la acción quedará registrada. Negamos la conclusión y añadimos $(\neg auditar)$. El conjunto de cláusulas es:

$$C_1 = (\neg admin \vee borrar), \quad C_2 = (\neg borrar \vee auditar), \quad C_3 = (admin), \quad C_4 = (\neg auditar)$$

Y la refutación va así, paso a paso:

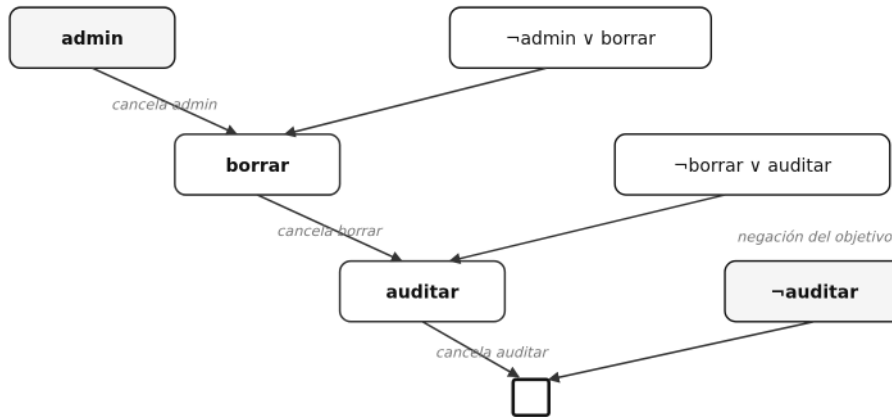
PASO	RESOLVEMOS	SOBRE EL LITERAL	RESOLVENTE
1	C_3 y C_1	<i>admin</i>	$(borrar)$
2	paso 1 y C_2	<i>borrar</i>	$(auditar)$
3	paso 2 y C_4	<i>auditar</i>	(vacía)

Hemos llegado a la cláusula vacía. La suposición «la acción no queda registrada» era incompatible con las premisas, así que la conclusión queda demostrada: la acción **quedará** registrada en auditoría. El árbol de esta refutación es el siguiente.

Refutación por resolución

Cada paso cancela un literal complementario; llegar a la cláusula vacía cierra la prueba.

CLÁUSULAS DE PARTIDA



☐ cuadro vacío = contradicción (cláusula vacía) al encontrarla, la conclusión queda probada

IA para gente curiosa / Facsímil 12 / Capítulo 01 / 686f6c61

Corrección y completitud

¿Por qué fiarse de este procedimiento? Por dos propiedades que cualquier mecanismo de inferencia debe ganarse:

- **Corrección** (en inglés *soundness*): la regla nunca miente. Todo lo que la resolución deriva es de verdad una consecuencia lógica de las premisas. Si llega a la cláusula vacía, el conjunto era de verdad insatisfacible. Una máquina correcta jamás demuestra algo falso.
- **Completitud** (*completeness*): la regla no se deja nada por demostrar. Si un conjunto de cláusulas es insatisfacible, la resolución es capaz de derivar la cláusula vacía a partir de él. Esto se conoce como **completitud refutacional**, y es el resultado profundo del trabajo de Robinson.

Juntas dicen algo extraordinario: con esta única regla, aplicada de forma mecánica, una máquina puede demostrar exactamente las consecuencias lógicas verdaderas, ni una más ni una menos. No necesita ingenio, no necesita elegir entre decenas de reglas. Necesita combinar cláusulas hasta encontrar la vacía. Esa es la promesa de la demostración automática de teoremas, y su raíz histórica está en el algoritmo de Davis, Putnam, Logemann y Loveland, anterior a la resolución y todavía vivo en el corazón de los solvers SAT.⁵

Lógica de primer orden: objetos, relaciones y cuantificadores

La lógica proposicional tiene un techo bajo. Imagina la frase «todos los administradores pueden borrar». En proposicional, «Ana es administradora», «Luis es administrador» y «Marta es administradora» son tres variables sin ninguna relación entre sí. No hay forma de

decir «todos» de golpe: tendrías que escribir una regla por persona, y si mañana entra alguien nuevo, tu lógica no sabe nada de ella. La estructura interna de los enunciados (que hablan de *objetos* que tienen *propiedades* y mantienen *relaciones*) se pierde por completo.

La **lógica de primer orden** (FOL) rompe ese techo. Distingue piezas dentro de cada enunciado:

PIEZA	QUÉ ES	EJEMPLO
Constantes	Objetos concretos del dominio.	<i>ana, factura9.</i>
Variables	Objetos cualesquiera, sin fijar.	<i>x, y.</i>
Predicados	Propiedades o relaciones, verdaderas o falsas.	<i>Admin(x), PuedeBorrar(x), Aprueba(x, y).</i>
Funciones	Aplicaciones que devuelven un objeto a partir de otros.	<i>jefe(x), madre(x).</i>
Cuantificadores	Alcance sobre los objetos.	$\forall$ («para todo»), $\exists$ («existe»).

Con esto, «todos los administradores pueden borrar» se escribe de una vez y vale para Ana, para Luis y para quien entre mañana:

$$\forall x (Admin(x) \rightarrow PuedeBorrar(x))$$

En palabras: sea quien sea ese objeto x , si resulta ser administrador entonces puede borrar; la regla vale para todos a la vez, también para quien entre mañana.

Y «existe al menos un administrador» se escribe:

$$\exists x Admin(x)$$

En palabras: hay como mínimo un objeto del que es cierto que es administrador, aunque no digamos cuál.

Los dos cuantificadores son duales, conectados por la negación de la misma forma que las leyes de De Morgan conectaban $\wedge$ y $\vee$: «no todos cumplen P » es lo mismo que «existe alguno que no cumple P ».

$$\neg \forall x P(x) \equiv \exists x \neg P(x) \qquad \neg \exists x P(x) \equiv \forall x \neg P(x)$$

En palabras: negar un «para todo» equivale a afirmar que existe al menos una excepción; negar un «existe» equivale a afirmar que la propiedad falla para todos sin excepción.

Esta capacidad expresiva es justo lo que hay debajo de las ontologías OWL y las lógicas de descripción del Facsímil 2: cuando declaras que «toda factura pertenece a exactamente un cliente», estás escribiendo una fórmula cuantificada de primer orden (restringida, para que el razonamiento siga siendo tratable). FOL es el lenguaje madre de toda esa familia.⁶

Skolemización: deshacerse de «existe»

La resolución trabaja con cláusulas, y las cláusulas no tienen cuantificadores explícitos. Para aplicar resolución en primer orden hay que preparar las fórmulas, y el paso más curioso es la **skolemización**: eliminar los cuantificadores existenciales sustituyendo la variable que afirman que existe por un objeto concreto.

La intuición es esta. Si digo $\exists x \text{ Admin}(x)$, «existe un administrador», puedo darle un nombre a ese administrador que sé que existe: lo llamo a_0 (una **constante de Skolem**) y reescribo $\text{Admin}(a_0)$. Si el «existe» está dentro de un «para todo» (cada empleado tiene un jefe: $\forall x \exists y \text{ Jefe}(y, x)$), el objeto que existe depende de x , así que en lugar de una constante uso una **función** de Skolem: $\forall x \text{ Jefe}(\text{jefe}(x), x)$. No estamos inventando información: solo le ponemos nombre a algo cuya existencia ya habíamos afirmado. Tras skolemizar, todos los cuantificadores son universales y se pueden dar por supuestos, y la fórmula se lleva a cláusulas como en proposicional.

Unificación: el motor del primer orden

Aquí está la pieza que de verdad distingue la resolución de primer orden. En proposicional, dos literales se cancelaban si uno era exactamente el complementario del otro. En primer orden, los literales tienen variables, y casi nunca coinciden palabra por palabra. $\text{Admin}(x)$ y $\text{Admin}(\text{ana})$ no son idénticos, pero está claro que el primero «encaja» con el segundo si hacemos que x sea ana . Encontrar ese encaje es la **unificación**: buscar una sustitución de variables que iguale dos expresiones.

Una **sustitución** se escribe como un conjunto de reemplazos, por ejemplo $\sigma = \{x/\text{ana}, y/\text{jefe}(\text{ana})\}$, que se lee «pon ana donde haya x y $\text{jefe}(\text{ana})$ donde haya y ». El algoritmo de unificación recorre las dos expresiones en paralelo:

CASO	QUÉ HACE
Dos constantes iguales	Siguen adelante; no hace falta sustituir nada.
Dos constantes distintas	Fallo: <i>ana</i> y <i>luis</i> no se pueden igualar.
Una variable y un término	Sustituye la variable por el término (si la variable no aparece dentro del término, lo que se llama <i>comprobación de ocurrencia</i>).
Dos predicados o funciones	Deben tener el mismo nombre y aridad; entonces unifica argumento a argumento.

Un ejemplo trabajado. Queremos unificar:

$$\text{Aprueba}(x, \text{factura}(x)) \quad \text{y} \quad \text{Aprueba}(\text{ana}, z)$$

Recorremos los argumentos. El primero: x frente a *ana*, variable contra constante, sustituimos $\{x/\text{ana}\}$. Aplicamos esa sustitución al resto, con lo que $\text{factura}(x)$ se vuelve $\text{factura}(\text{ana})$. El segundo argumento: $\text{factura}(\text{ana})$ frente a z , sustituimos $\{z/\text{factura}(\text{ana})\}$. El resultado es el **unificador**:

$$\sigma = \{x/\text{ana}, z/\text{factura}(\text{ana})\}$$

Aplicado a ambos literales, los dos se vuelven $\text{Aprueba}(\text{ana}, \text{factura}(\text{ana}))$: idénticos. El algoritmo siempre encuentra, si existe, el **unificador más general**, el que sustituye lo mínimo imprescindible y deja todo lo demás libre. Esa generalidad es lo que hace que una sola derivación valga para infinitos casos concretos.

El recorrido completo de ese ejemplo, argumento a argumento, es el siguiente.

Unificación: caminar los dos términos en paralelo

Mismo nombre y misma aridad: se comparan los argumentos uno a uno y se acumulan las sustituciones.

LITERAL A

Aprueba(x, factura(x))

LITERAL B

Aprueba(ana, z)

1. argumento 1: x (variable) frente a ana (constante) → sustituye {x / ana}

2. aplica {x / ana} al resto: factura(x) pasa a ser factura(ana)

3. argumento 2: factura(ana) (término) frente a z (variable) → sustituye {z / factura(ana)}

$\sigma = \{ x / ana, z / factura(ana) \}$
unificador más general

Aprueba(ana, factura(ana))
los dos literales, ya idénticos

IA para gente curiosa / Facsimil 1.2 / Capítulo 01 / 686f6c61

Resolución de primer orden

Con la unificación, la regla de resolución se generaliza casi sola: para cancelar dos literales complementarios, primero los unificamos y luego aplicamos la sustitución al resolvente. El ejemplo canónico, con más de dos mil años de antigüedad, es la mortalidad de Sócrates:

- «Todos los humanos son mortales»: $\forall x (Humano(x) \rightarrow Mortal(x))$, que en cláusula es $(\neg Humano(x) \vee Mortal(x))$.
- «Sócrates es humano»: $(Humano(socrates))$.
- Queremos probar: $Mortal(socrates)$. Negamos: $(\neg Mortal(socrates))$.

Resolvemos. La cláusula $(\neg Humano(x) \vee Mortal(x))$ y $(Humano(socrates))$ se unifican con $\sigma = \{x/socrates\}$, cancelando *Humano*, y dan $(Mortal(socrates))$. Esta y $(\neg Mortal(socrates))$ se cancelan y dan la cláusula vacía. Sócrates es mortal. La estructura es idéntica a la refutación proposicional de antes; lo único nuevo es que la unificación ha hecho el trabajo de emparejar la regla general con el caso concreto.

Cláusulas de Horn y la conexión con Prolog

La resolución de primer orden es completa, pero buscar a ciegas entre todas las combinaciones de cláusulas es caro. Resulta que si restringimos un poco la forma de las cláusulas, el razonamiento se vuelve mucho más manejable y, además, se parece muchísimo a programar.

Una **cláusula de Horn** es una cláusula con **como mucho un literal positivo**. Suena técnico, pero tiene una lectura cotidiana preciosa. Una cláusula de Horn con exactamente un positivo, como $(\neg A \vee \neg B \vee C)$, es equivalente a la implicación $(A \wedge B) \rightarrow C$: «si se cumplen A y B, entonces C». Es decir, una regla. Y eso es exactamente lo que escribes cuando programas en Prolog, solo que del revés:

```
% C :- A, B.    se lee "C es cierto si A y B lo son"
puede_borrar(X) :- administrador(X), cuenta_activa(X).

administrador(ana).
cuenta_activa(ana).

% Pregunta:
?- puede_borrar(ana).
% Prolog responde: true
```

Prolog es, en el fondo, un motor de resolución sobre cláusulas de Horn.⁷ Cuando le preguntas `?- puede_borrar(ana)`, niega el objetivo (como en la refutación) e intenta derivar la cláusula vacía encadenando reglas hacia atrás: para probar `puede_borrar(ana)` necesita `administrador(ana)` y `cuenta_activa(ana)`, que están como hechos, así que cierra la prueba. Esta estrategia dirigida por el objetivo se llama **resolución SLD**, y es lo que hace que Prolog sea a la vez una lógica y un lenguaje de programación: el mismo texto se lee como especificación declarativa («qué es verdad») y como programa («cómo calcularlo»).

La teoría que garantiza que esto funciona (que el significado lógico de un programa Prolog y su comportamiento al ejecutarse coinciden) es la base de la programación lógica.⁸ Y hay un bono de eficiencia: decidir la satisfacibilidad de un conjunto de cláusulas de Horn (el problema HORN-SAT) se resuelve en tiempo lineal, no exponencial. Restringirse a Horn nos saca de la zona difícil de SAT que vimos en el Facsímil 2 y nos pone en terreno rápido y predecible. Por eso los motores de reglas, muchos sistemas expertos (que veremos en el capítulo 4 de este facsímil) y buena parte de la programación lógica viven cómodamente en este fragmento.

Los límites: qué puede y qué no puede una máquina lógica

Toda esta potencia tiene fronteras, y conocerlas es parte de usar la lógica con honestidad.

Decidibilidad. La lógica proposicional es **decidible**: existe un procedimiento que, para cualquier fórmula, termina siempre y responde sí o no (la tabla de verdad, o un solver SAT). Puede ser lento (NP-completo), pero acaba. La lógica de primer orden, en cambio, es solo **semidecidible**. Si una fórmula es de verdad una consecuencia lógica, la resolución terminará encontrando la prueba; pero si **no** lo es, el procedimiento puede no detenerse nunca, siguiendo generando resolventes sin llegar a conclusión. No es un defecto de implementación: se demostró que no puede existir ningún algoritmo que decida siempre la validez en primer orden. Hay un límite teórico, no de ingeniería.

Explosión combinatoria. Incluso dentro de lo decidible, el número de cláusulas que se pueden generar crece de forma brutal. La unificación produce resolventes nuevos constantemente, y sin estrategias de control un demostrador se ahoga en su propia

producción. La investigación en demostración automática es, en buena medida, la búsqueda de heurísticas para no explorar de más.

El frame problem. Cuando se intentó usar la lógica para que un agente razonara sobre acciones en el mundo (si muevo esta caja, ¿qué cambia y qué sigue igual?), apareció un problema sutil: hace falta declarar explícitamente todo lo que **no** cambia con cada acción, y eso es inabarcable.⁹ Este *frame problem* fue uno de los grandes quebraderos de cabeza de la IA simbólica clásica y una de las razones por las que, para ciertos dominios, se buscaron enfoques distintos. Lo mencionamos porque enmarca por qué la lógica pura no se quedó como la única herramienta: convive con la probabilidad (el próximo capítulo) y con el aprendizaje.

En el día a día

Aunque no escribas $\forall$ ni la cláusula vacía en tu trabajo, esta maquinaria está funcionando por debajo de cosas que usas a diario.

Cada vez que un sistema de permisos decide si puedes hacer una acción, evalúa reglas: «si el rol es administrador y la cuenta está activa, permitir». Eso son cláusulas de Horn evaluándose. Cada vez que un validador comprueba que una salida cumple un esquema (un JSON con los campos obligatorios, los tipos correctos, las restricciones de cardinalidad), está aplicando lógica determinista, no plausibilidad. Cada vez que una ontología responde «esta entidad es un documento fiscal porque es una factura, y toda factura lo es», ha hecho una inferencia por subsunción, que es resolución disfrazada.

En el desarrollo de software, los verificadores formales que comprueban que un programa cumple su especificación usan demostradores SMT como Z3, que combinan resolución con teorías (aritmética, arrays, cadenas). Los compiladores optimizan usando análisis lógico. Y en sistemas con LLMs, la lógica es la red de seguridad: el modelo propone, pero una capa lógica verificable decide si la propuesta se ejecuta.

Por qué debería importarte

Porque la diferencia entre «suena correcto» y «es correcto» es exactamente la diferencia entre un modelo generativo y un demostrador lógico. Un LLM te puede dar una respuesta convincente y equivocada con total seguridad. Un razonador lógico correcto nunca afirma algo falso a partir de premisas verdaderas: si dice que algo se sigue, se sigue.

Esa garantía es escasa y valiosa. En todo el resto del facsímil aparecen sistemas que mezclan generación con verificación, y la parte de verificación, la que de verdad protege tus decisiones, casi siempre tiene una lógica determinista debajo. Entender resolución, refutación y la diferencia entre validez y satisfacibilidad te da el vocabulario para saber dónde poner una

garantía dura y dónde basta con una sugerencia blanda. Es la misma lección del Facsímil 2: cuando hay reglas que no se pueden violar, la aceptación no debe depender de si la respuesta suena bien.

Dónde solía tropezar yo

ERROR	POR QUÉ ES UN ERROR	ANTÍDOTO
Confundir validez con satisfacibilidad	Una fórmula satisfacible solo es «no imposible». Una válida es «siempre verdadera». Tratar la primera como la segunda lleva a aceptar cosas que solo se cumplen a veces.	Pregúntate: ¿quiero que se cumpla <i>siempre</i> (validez) o que sea <i>posible</i> (satisfacibilidad)? Recuerda el puente: válida equivale a que su negación sea insatisfacible.
Leer mal la implicación material	$A \rightarrow B$ es verdadera cuando A es falsa. Esperar que «si llueve, llevo paraguas» diga algo cuando no llueve es esperar de más.	Quédate con la única fila falsa: A verdadero y B falso. Solo ahí se rompe la promesa.
Intentar demostrar ψ directamente	Derivar la conclusión a pelo es difícil y poco sistemático.	Usa refutación: niega la conclusión, pásalo todo a CNF y busca la cláusula vacía. Es lo que hace una máquina.
Olvidar la comprobación de ocurrencia al unificar	Unificar x con $factura(x)$ crea un término infinito y rompe el algoritmo.	Antes de sustituir x por un término, comprueba que x no aparece dentro de ese término.
Creer que FOL siempre termina	Es solo semidecidible: si la conclusión no se sigue, el demostrador puede no parar nunca.	Pon límites de tiempo o de profundidad, y desconfía de un «todavía pensando» eterno.

Cómo encaja todo

Este capítulo parece un rincón teórico, pero en realidad es el suelo sobre el que se apoyan media docena de cosas que ya viste o que vas a ver en el resto del facsímil. Vale la pena seguir los hilos uno a uno, porque cada uno explica por qué la lógica no es una curiosidad histórica, sino una pieza que sigue trabajando por debajo.

El hilo más directo va al **Facsímil 2, capítulo 5**, donde aparecieron los solvers SAT. Allí la pregunta era «¿hay alguna forma de asignar valores que cumpla todas estas restricciones a la vez?»: cuadrar un horario sin solapes, colocar las piezas de un sudoku, repartir turnos respetando las reglas. Esa pregunta es, palabra por palabra, la **satisfacibilidad** de este

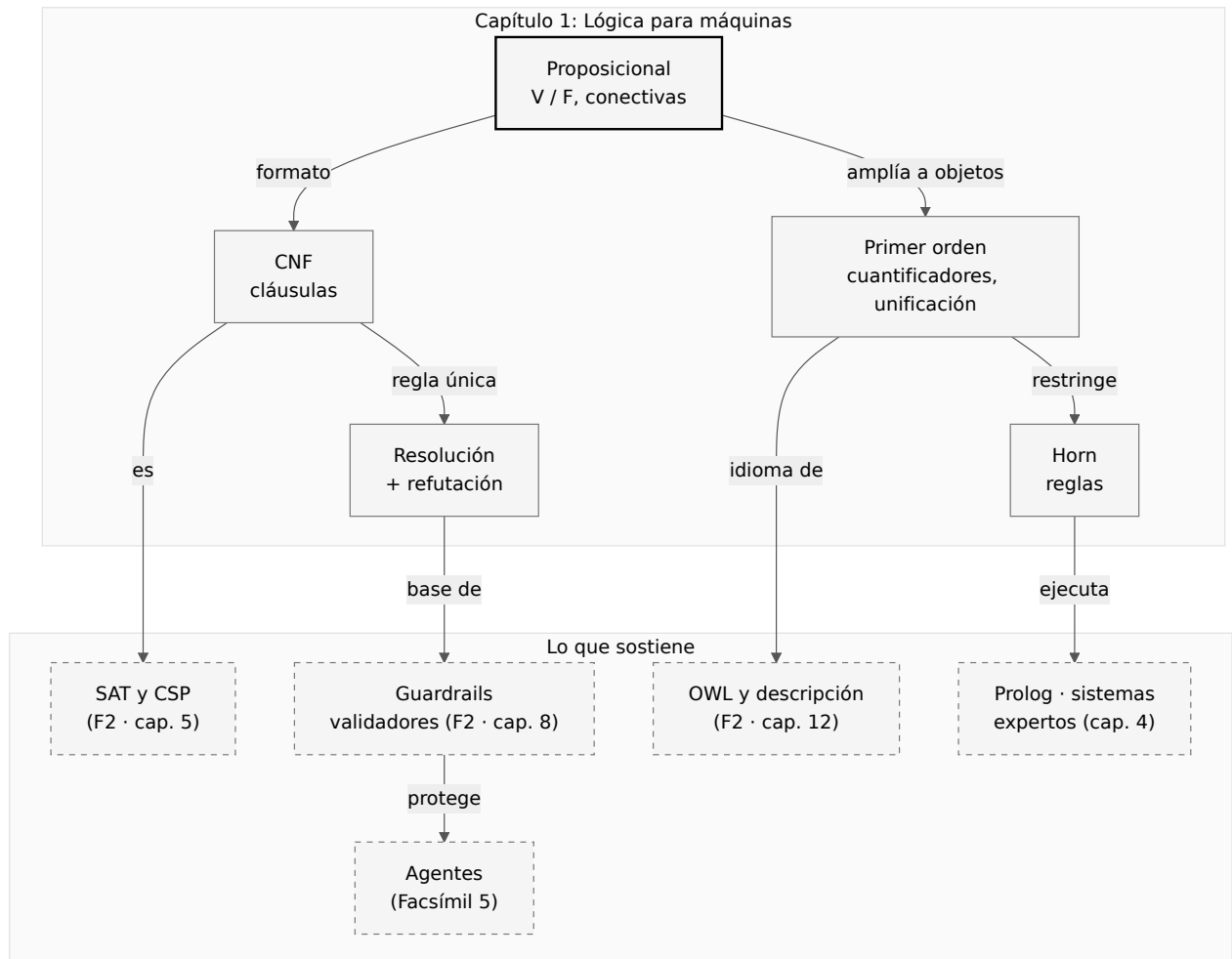
capítulo. Lo que cambia es el envoltorio: lo que el Facsímil 2 contaba como un problema de búsqueda, aquí se ve como lo que es por dentro, una fórmula en CNF a la que le buscamos un modelo. Y el puente «validez equivale a insatisfacibilidad de la negación» es justo lo que permite usar ese mismo solver no solo para encajar restricciones, sino para demostrar teoremas: le das la vuelta a la pregunta y la máquina te dice si algo se cumple siempre.

El segundo hilo lleva al **Facsímil 2, capítulo 8**, el de las restricciones como guardrails. Allí la idea era poner barreras duras alrededor de un sistema que genera texto: reglas que no son sugerencias, sino verjas que aceptan o rechazan una salida sin negociar. Eso es exactamente la resolución y la refutación funcionando como portero. Piensa en un asistente que redacta un correo a un cliente: el modelo propone el texto, pero antes de enviarlo una capa de reglas comprueba cosas que no pueden fallar, como que no haya un importe sin formato válido o que no se prometa un descuento no autorizado. Esa comprobación no «suena bien o mal», se cumple o no se cumple, y debajo hay cláusulas evaluándose con la maquinaria de aquí.

El tercer hilo, también en el **Facsímil 2, capítulo 12**, son las ontologías OWL y las lógicas de descripción. Cuando declaras que «toda factura pertenece a exactamente un cliente» o que «todo documento fiscal es un documento», estás escribiendo fórmulas de primer orden con cuantificadores, solo que con una sintaxis amable y recortada para que el razonamiento siga siendo rápido. Por eso, cuando una ontología deduce sola que una entidad concreta es un documento fiscal porque es una factura, no está adivinando: está haciendo la misma inferencia que vimos con Sócrates, una regla general que se aplica a un caso concreto mediante unificación. La lógica de descripción es primer orden domesticado, y este capítulo es su lengua materna.

El cuarto hilo salta al **Facsímil 5**, el de los agentes. Un agente que actúa en el mundo necesita saber qué tiene permitido hacer antes de hacerlo, y esos permisos no son intuiciones: son reglas. «Si el rol es administrador y la cuenta está activa, puede borrar» es una cláusula de Horn, idéntica a las que escribíamos en Prolog. Cada vez que el agente se plantea una acción, evalúa esas reglas y la respuesta es un sí o un no comprobable. La diferencia entre un agente en el que confías y uno que da miedo soltar suele estar justo ahí: en si sus permisos son lógica determinista o una corazonada del modelo.

Y el último hilo se queda en casa, en el **capítulo 4 de este mismo facsímil**, el del motor de inferencia. Lo que aquí hemos descrito como reglas y como una regla de resolución que se aplica en bucle, allí se convierte en la máquina que de verdad las ejecuta: un motor que toma hechos y reglas y va derivando conclusiones, encadenando hacia delante o hacia atrás como hace Prolog. Este capítulo te da el vocabulario (cláusula, resolvente, unificación, cláusula vacía); el capítulo 4 te enseña el motor que los pone a girar. Conviene leerlos seguidos, porque uno es la gramática y el otro, la conversación.



Vocabulario aprendido

TÉRMINO	DEFINICIÓN
Lógica proposicional	Lógica de enunciados que son verdaderos o falsos, combinados con conectivas.
Conectiva lógica	Operador que combina enunciados: negación, conjunción, disyunción, implicación y bicondicional.
Tabla de verdad	Tabla que da el valor de una fórmula para cada combinación de valores de sus variables.
Validez	Una fórmula es válida si es verdadera en toda interpretación (tautología).
Satisfacibilidad	Una fórmula es satisfacible si existe al menos una interpretación que la hace verdadera.
Forma normal conjuntiva	Conjunción de cláusulas, donde cada cláusula es una disyunción de literales.
Modus ponens	Regla de inferencia: de A y $A \rightarrow B$ se deriva B .
Resolución	Regla que, de dos cláusulas con un literal complementario, deriva una nueva cláusula.
Refutación	Probar que algo se deduce demostrando que su negación lleva a contradicción (cláusula vacía).
Lógica de primer orden	Lógica con predicados, funciones, variables y cuantificadores sobre objetos.
Cuantificador	Símbolo que expresa «para todo» ($\forall$) o «existe» ($\exists$).
Unificación	Proceso de encontrar una sustitución que iguala dos expresiones lógicas.
Cláusula de Horn	Cláusula con como mucho un literal positivo; base de Prolog y la programación lógica.
Compleitud	Una regla de inferencia es completa si puede derivar toda consecuencia lógica verdadera.

Antes de pasar página

☐ ¿Sé construir la tabla de verdad de una fórmula con dos o tres variables y clasificarla? (Si no, vuelve a «Tablas de verdad, tautologías y la diferencia entre validez y satisfacibilidad».)

- ☐ ¿Distingo validez de satisfacibilidad y entiendo el puente entre ambas por la negación? (Si no, vuelve a la misma sección.)
- ☐ ¿Puedo pasar una implicación a CNF en tres pasos? (Si no, vuelve a «Forma normal conjuntiva».)
- ☐ ¿Sé aplicar la regla de resolución y explicar qué es la cláusula vacía? (Si no, vuelve a «La regla de resolución».)
- ☐ ¿Entiendo cómo una refutación demuestra una conclusión llegando a la cláusula vacía? (Si no, vuelve a «Demostrar por refutación».)
- ☐ ¿Veo por qué FOL expresa «todos» y «existe» y la proposicional no? (Si no, vuelve a «Lógica de primer orden».)
- ☐ ¿Puedo unificar dos literales y dar el unificador más general? (Si no, vuelve a «Unificación: el motor del primer orden».)
- ☐ ¿Reconozco una cláusula de Horn como una regla y su relación con Prolog? (Si no, vuelve a «Cláusulas de Horn y la conexión con Prolog».)

En resumen

IDEA FUERZA	DETALLE
La lógica razona por la forma, no por el contenido.	Por eso una conclusión válida es mecanizable: una máquina la comprueba sin entender el tema.
Validez e insatisfacibilidad son dos caras.	Demostrar que algo es siempre verdadero equivale a demostrar que su negación es imposible.
La resolución es una sola regla, correcta y completa.	Aplicada por refutación hasta la cláusula vacía, demuestra cualquier consecuencia lógica.
El primer orden añade objetos y cuantificadores.	Con unificación, una regla general se aplica a infinitos casos; es el idioma de OWL y Prolog.
Toda esta lógica sostiene la IA verificable del facsímil.	SAT, guardrails, ontologías y sistemas expertos descansan en proposicional, resolución y Horn.

Para saber más

Cook, S. A. (1971). The complexity of theorem-proving procedures. En *Proceedings of the Third Annual ACM Symposium on Theory of Computing* (pp. 151-158). ACM.

<https://doi.org/10.1145/800157.805047>

- Davis, M., Logemann, G. y Loveland, D. (1962). A machine program for theorem-proving. *Communications of the ACM*, 5(7), 394-397. <https://doi.org/10.1145/368273.368557>
- Genesereth, M. R. y Nilsson, N. J. (1987). *Logical foundations of artificial intelligence*. Morgan Kaufmann.
- Huth, M. y Ryan, M. (2004). *Logic in computer science: modelling and reasoning about systems* (2.^a ed.). Cambridge University Press.
- Kowalski, R. (1979). *Logic for problem solving*. North-Holland.
- Lloyd, J. W. (1987). *Foundations of logic programming* (2.^a ed.). Springer-Verlag.
- Robinson, J. A. (1965). A machine-oriented logic based on the resolution principle. *Journal of the ACM*, 12(1), 23-41. <https://doi.org/10.1145/321250.321253>
- Russell, S. y Norvig, P. (2021). *Artificial intelligence: a modern approach* (4.^a ed.). Pearson.
-

Notas

1. Huth, M. y Ryan, M. (2004). *Logic in computer science: modelling and reasoning about systems* (2.^a ed.). Cambridge University Press. El capítulo inicial introduce con rigor la sintaxis y la semántica de la lógica proposicional, y es una de las exposiciones más limpias para informáticos. ↵
2. Cook, S. A. (1971). The complexity of theorem-proving procedures. En *Proceedings of the Third Annual ACM Symposium on Theory of Computing* (pp. 151-158). ACM. <https://doi.org/10.1145/800157.805047> El resultado de Cook es el que conecta la lógica proposicional con la teoría de la complejidad y justifica por qué la fuerza bruta de la tabla de verdad no escala. ↵
3. Robinson, J. A. (1965). A machine-oriented logic based on the resolution principle. *Journal of the ACM*, 12(1), 23-41. <https://doi.org/10.1145/321250.321253> Este artículo es el origen de la resolución; el propio título («una lógica orientada a máquinas») deja claro que el objetivo era la demostración automática, no la elegancia para el lector humano. ↵
4. Genesereth, M. R. y Nilsson, N. J. (1987). *Logical foundations of artificial intelligence*. Morgan Kaufmann. Su tratamiento de la resolución como procedimiento de refutación es el estándar en los textos de IA simbólica y conecta la regla con la representación del conocimiento. ↵
5. Davis, M., Logemann, G. y Loveland, D. (1962). A machine program for theorem-proving. *Communications of the ACM*, 5(7), 394-397. <https://doi.org/10.1145/368273.368557> El procedimiento DPLL, base de los solvers SAT modernos, comparte con la resolución la idea de razonar sobre cláusulas en CNF para decidir la insatisfacibilidad. ↵

6. Genesereth, M. R. y Nilsson, N. J. (1987). *Logical foundations of artificial intelligence*. Morgan Kaufmann. El libro desarrolla la lógica de primer orden precisamente como lenguaje de representación del conocimiento para la inteligencia artificial. ↵
7. Kowalski, R. (1979). *Logic for problem solving*. North-Holland. Kowalski formuló la idea de que la lógica de Horn puede leerse a la vez como especificación declarativa y como programa ejecutable, el principio que dio nacimiento a la programación lógica. ↵
8. Lloyd, J. W. (1987). *Foundations of logic programming* (2.ª ed.). Springer-Verlag. Es el tratado de referencia sobre la semántica de los programas lógicos y la corrección de la resolución SLD sobre cláusulas de Horn. ↵
9. Russell, S. y Norvig, P. (2021). *Artificial intelligence: a modern approach* (4.ª ed.). Pearson. Los capítulos dedicados a la lógica proposicional, de primer orden y a la inferencia tratan la decidibilidad, la resolución y el frame problem como temas centrales de la representación del conocimiento. ↵

Cuadernos para practicar

Escanea el código con el móvil para abrir el cuaderno en Google Colab y ejecutarlo sin instalar nada.



Lógica y resolución: razonar con reglas, no con datos

<https://colab.research.google.com/github/686f6c61/iaparagentecuriosa-notebooks/blob/main/Facs%3ADmil%2012%2F01-logica-y-resolucion.ipynb>

Capítulo 02: Razonamiento bajo incertidumbre: Bayes y redes bayesianas

Entrando en el tema

En el capítulo anterior, una máquina razonaba con verdades y falsedades. Una proposición era verdadera o falsa; una regla se disparaba o no se disparaba; una contradicción cerraba el caso. Esa disciplina es preciosa cuando el mundo se deja describir con afirmaciones tajantes. El problema es que casi nunca lo hace.

Piensa en una frase tan corriente como “la prueba ha dado positiva”. La lógica del capítulo 1 quiere convertirla en “el paciente está enfermo: verdadero”. Pero esa traducción es falsa, y de una forma peligrosa. Un positivo no implica enfermedad: la sugiere, la hace más probable, mueve la creencia en una dirección. Cuánto la mueve depende de cosas que la lógica binaria no sabe representar: cómo de frecuente es la enfermedad, cómo de fiable es la prueba, cuántas falsas alarmas produce. Si fuerzas el mundo a entrar en verdadero/falso, pierdes justo la información que necesitas para decidir bien.

Este capítulo cambia la herramienta. En lugar de tratar cada hecho como verdadero o falso, le asignamos un número entre 0 y 1 que mide nuestro **grado de creencia**: 0 es “imposible”, 1 es “seguro”, y todo lo interesante vive en medio. Esa idea, la probabilidad entendida como creencia razonada y no solo como frecuencia de un experimento, es la que permite a una máquina razonar cuando los hechos son inciertos.¹ La lógica no se tira a la basura: se generaliza. La verdad y la falsedad pasan a ser los dos extremos de una escala continua.

Y no es un parche improvisado. La probabilidad tiene reglas tan estrictas como la lógica, y con dos herramientas (el teorema de Bayes para actualizar creencias y las redes bayesianas para organizarlas) se convierte en una maquinaria de razonamiento completa. Es la misma maquinaria que, por debajo, sostiene buena parte del aprendizaje automático del Facsímil 1 y la calibración de modelos del Facsímil 7. Empecemos por las reglas.

El grado de creencia tiene reglas

Si vamos a representar la incertidumbre con números, esos números no pueden ser arbitrarios. La teoría de la probabilidad descansa sobre tres axiomas, propuestos por Kolmogórov, que cualquier asignación de creencias coherente debe cumplir. Son pocos y sencillos:

AXIOMA	ENUNCIADO	LECTURA COTIDIANA
No negatividad	$P(A) \geq 0$	Ninguna creencia puede ser “menos que imposible”.
Normalización	$P(\Omega) = 1$	Algo del conjunto de resultados posibles ocurre con certeza.
Aditividad	Si A y B son incompatibles, $P(A \vee B) = P(A) + P(B)$	Las probabilidades de sucesos que no pueden coincidir se suman.

De ahí sale todo lo demás. Por ejemplo, como un suceso A y su negación $\neg A$ son incompatibles y juntos cubren todo lo posible, se deduce $P(A) + P(\neg A) = 1$. Esa pequeña igualdad ya es más de lo que la lógica binaria te daba: te dice que creer al 70 % en algo es exactamente creer al 30 % en lo contrario.

La pieza que de verdad enciende el razonamiento es la **probabilidad condicional**. Mide cuánto crees en A una vez que sabes que B ha ocurrido:

$$P(A | B) = \frac{P(A \wedge B)}{P(B)}, \quad P(B) > 0$$

SÍMBOLO	SIGNIFICADO	EJEMPLO
$P(A B)$	Creencia en A sabiendo B .	Probabilidad de estar enfermo dado un positivo.
$P(A \wedge B)$	Probabilidad de que ocurran ambos.	Estar enfermo y dar positivo a la vez.
$P(B)$	Probabilidad de la condición observada.	Probabilidad de dar positivo, sin más datos.

En palabras: la probabilidad de A sabiendo B es la fracción de los casos en que ocurre B que, además, cumplen A . Es lo mismo que preguntar «de la gente que ha dado positivo, ¿qué parte está enferma?».

La fórmula dice algo muy intuitivo: condicionar es **reducir el universo**. Antes mirabas todos los casos posibles; al saber que B ocurre, te quedas solo con la franja en la que B es cierto y vuelves a medir dentro de ella qué proporción cumple también A . Dividir por $P(B)$ es justamente reescalar esa franja para que vuelva a sumar 1.

Reordenando la definición aparece la **regla del producto**, que descompone la probabilidad conjunta en pasos encadenados:

$$P(A \wedge B) = P(A | B) P(B) = P(B | A) P(A)$$

SÍMBOLO	SIGNIFICADO	EJEMPLO
$P(A \wedge B)$	Probabilidad de que ocurran ambos sucesos.	Que llueva y que hayas cogido el paraguas.
$P(A B) P(B)$	Descomposición fijando B primero.	Probabilidad de paraguas si llueve, por la de que llueva.
$P(B A) P(A)$	La misma cantidad fijando A primero.	Probabilidad de lluvia si hay paraguas, por la de paraguas.

En palabras: la probabilidad de que dos cosas pasen a la vez es la de una de ellas multiplicada por la de la otra una vez supuesta la primera; y da igual cuál tomes como punto de partida, el resultado es el mismo.

Y de la regla del producto, aplicada a muchas variables, sale la **regla de la cadena**, que escribiremos en un momento porque es la columna vertebral de las redes bayesianas:

$$P(X_1, X_2, \dots, X_n) = \prod_{i=1}^n P(X_i | X_1, \dots, X_{i-1})$$

SÍMBOLO	SIGNIFICADO	EJEMPLO
$X_1, \dots, X_n$	Las variables, en un orden cualquiera que fijas tú.	Las palabras de una frase, de izquierda a derecha.
$\prod_{i=1}^n$	Producto de los n factores.	Multiplicar los términos uno tras otro.
$P(X_i X_1, \dots, X_{i-1})$	Probabilidad de cada variable dadas todas las anteriores.	La de cada palabra dado lo ya escrito.

En palabras: la probabilidad conjunta de muchas variables se monta en cadena: eliges un orden y vas multiplicando la probabilidad de cada una condicionada a todas las que ya han salido.

Falta una operación más, la **marginalización**, que hace lo contrario de condicionar: en lugar de fijar una variable, la elimina sumando sobre todos sus valores. Si conoces la conjunta de A y B pero solo te interesa A , barres B :

$$P(A) = \sum_b P(A \wedge B=b) = \sum_b P(A | B=b) P(B=b)$$

SÍMBOLO	SIGNIFICADO	EJEMPLO
$P(A)$	Probabilidad de A sola, sin condicionar a nada.	Probabilidad de dar positivo, mires a quien mires.
$\sum_b$	Suma sobre todos los valores posibles de B .	Sumar el caso «enfermo» y el caso «sano».
$P(A B=b) P(B=b)$	Lo que aporta cada valor de B a A .	Positivos de los enfermos más positivos de los sanos.

En palabras: para quitarte de encima una variable que no te interesa, recorres todos sus valores posibles y sumas la probabilidad de A en cada caso, pesando cada uno por lo probable que era ese valor.

Esta suma se llama también **regla de la probabilidad total**, y es la que usaremos enseguida para calcular la probabilidad de “dar positivo” sumando las dos formas de llegar a un positivo: estando enfermo y no estándolo. Con condicional, producto y marginalización ya tenemos toda el álgebra que necesita el teorema de Bayes.

El teorema de Bayes: dar la vuelta a la pregunta

A menudo sabemos calcular una dirección de la probabilidad condicional, pero nos interesa la contraria. Un médico sabe, por estudios clínicos, con qué frecuencia una persona enferma da positivo en una prueba: eso es $P(\text{positivo} | \text{enfermo})$. Pero el paciente que tiene el positivo en la mano no quiere saber eso. Quiere saber $P(\text{enfermo} | \text{positivo})$: la pregunta al revés. El teorema de Bayes es exactamente el puente entre ambas direcciones.²

Se deduce en una línea igualando las dos formas de la regla del producto, $P(H \wedge E) = P(E | H)P(H) = P(H | E)P(E)$, y despejando:

$$P(H | E) = \frac{P(E | H) P(H)}{P(E)}$$

Cada pieza tiene un nombre y un papel, y conviene memorizarlos porque organizan toda la forma de pensar:

PIEZA	NOMBRE	QUÉ REPRESENTA
$P(H)$	Probabilidad previa	Lo que creías sobre la hipótesis H antes de ver nada.
$P(E H)$	Verosimilitud	Cómo de esperable es la evidencia E si H fuese cierta.
$P(E)$	Evidencia	Cómo de esperable es E en total, sea cual sea la causa.
$P(H E)$	Probabilidad posterior	Lo que crees sobre H después de incorporar E .

En palabras: tu nueva creencia en la hipótesis es la que ya tenías, multiplicada por cuánto encaja la evidencia con ella y rebajada por cuánto era de esperar esa evidencia en cualquier caso. Si la evidencia era esperable hasta sin la hipótesis, mueve poco; si solo la hipótesis la explicaba, mueve mucho.

El término del denominador, la evidencia, casi nunca se da directamente: se calcula marginalizando sobre las hipótesis posibles. Si H y $\neg H$ agotan los casos:

$$P(E) = P(E | H) P(H) + P(E | \neg H) P(\neg H)$$

En palabras: la probabilidad total de ver la evidencia es la suma de las dos formas de llegar a ella: que la hipótesis sea cierta y produzca la evidencia, o que sea falsa y aun así la evidencia aparezca (la falsa alarma).

Leído despacio, el teorema cuenta una historia: partes de una creencia (previa), la multiplicas por cuánto encaja la evidencia con esa creencia (verosimilitud) y la divides por cuánto era de esperar la evidencia en general (evidencia) para obtener tu creencia actualizada (posterior). La probabilidad posterior de hoy es la previa de mañana: cada nuevo dato vuelve a entrar en la misma maquinaria. Esa es la esencia del razonamiento bajo incertidumbre.

El ejemplo que casi todo el mundo falla

Vamos con el caso clásico, porque su resultado es tan contraintuitivo que conviene calcularlo a mano y no fiarse del instinto. Una enfermedad afecta al **1 %** de la población. Existe una prueba con estas características:

- Si una persona está enferma, la prueba da positivo el **99 %** de las veces (sensibilidad alta).
- Si una persona está sana, la prueba da positivo igualmente el **5 %** de las veces (falsos positivos).

Una persona se hace la prueba y da positivo. Pregunta directa: ¿qué probabilidad tiene de estar realmente enferma? El instinto de casi todo el mundo, incluidos muchos médicos, dispara una cifra alta, en torno al 95 %, confundiendo la fiabilidad de la prueba con la probabilidad de enfermedad. Hagamos las cuentas de verdad.

Primero, traduzcamos los datos al lenguaje de Bayes. Sea H = “estar enfermo” y E = “dar positivo”:

DATO DEL ENUNCIADO	NOTACIÓN	VALOR
Prevalencia (previa)	$P(H)$	0,01
Sano (probabilidad previa complementaria)	$P(\neg H)$	0,99
Sensibilidad (verosimilitud)	$P(E H)$	0,99
Falsos positivos	$P(E \neg H)$	0,05

Calculamos la evidencia, es decir, la probabilidad total de dar positivo, sumando las dos maneras de obtener un positivo:

$$P(E) = P(E | H) P(H) + P(E | \neg H) P(\neg H)$$

$$P(E) = (0,99)(0,01) + (0,05)(0,99) = 0,0099 + 0,0495 = 0,0594$$

Ahora aplicamos Bayes:

$$P(H | E) = \frac{P(E | H) P(H)}{P(E)} = \frac{0,0099}{0,0594} = 0,1667$$

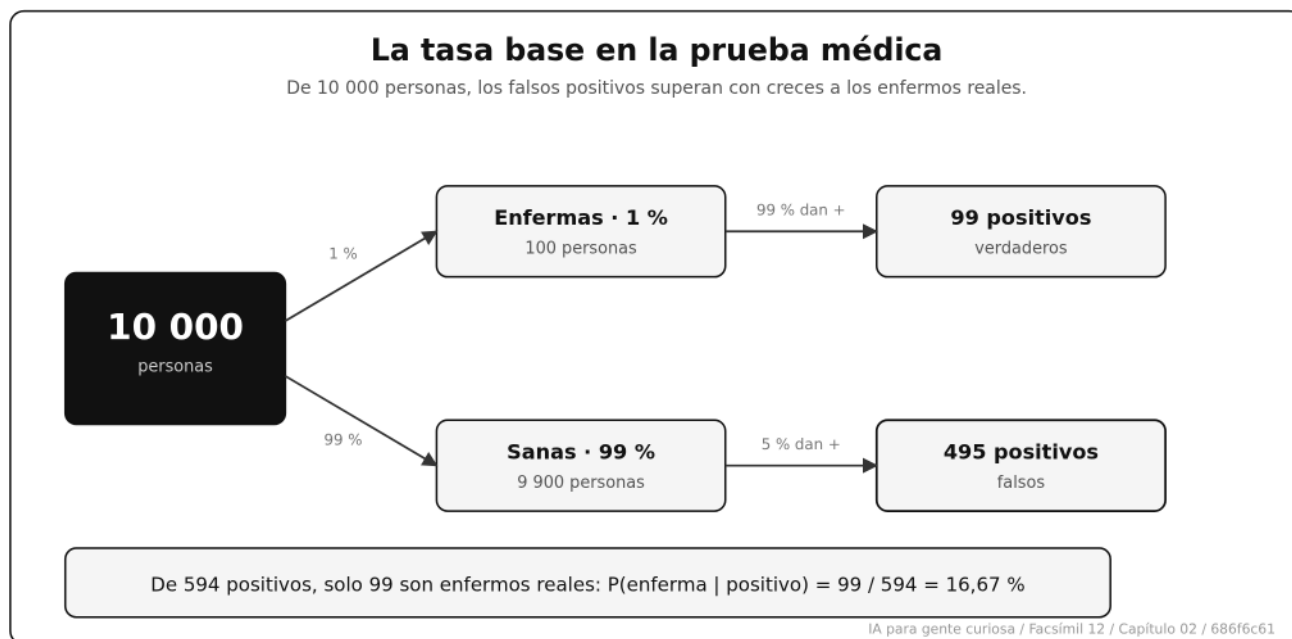
El resultado es **16,67 %**, no el 95 % que sugería el instinto. Una persona con un positivo en esta prueba tiene, todavía, una probabilidad de cinco sextos de estar sana. ¿Por qué? Porque la enfermedad es tan rara que los falsos positivos, ese 5 % aplicado al enorme 99 % de personas sanas, generan muchísimos más positivos que los verdaderos enfermos. Mira las cifras crudas sobre 10 000 personas:

GRUPO	PERSONAS	DAN POSITIVO	CÓMO SE OBTIENE
Enfermas	100	99	$10\,000 \times 0,01 \times 0,99$
Sanas	9 900	495	$10\,000 \times 0,99 \times 0,05$
Total positivos		594	99 verdaderos + 495 falsos

De los 594 positivos, solo 99 son personas realmente enfermas: $99/594 = 0,1667$, el mismo 16,67 %. Esa es la lección de la **tasa base**: ignorar lo raro que es una hipótesis previa lleva a sobreinterpretar la evidencia. Es el mismo error que cometería un detector de fraude que

confunde su precisión con la probabilidad real de fraude, el problema de clases desbalanceadas que vimos en el Facsímil 1, capítulo 11. La probabilidad previa no es un adorno: es la mitad del cálculo.

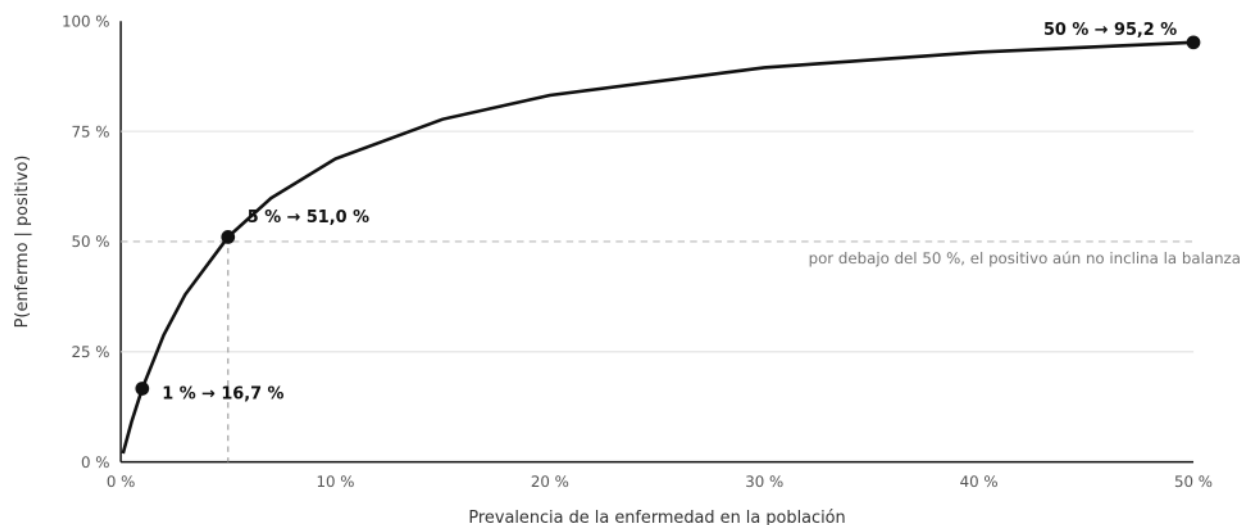
El dibujo deja a la vista por qué el instinto se equivoca: el grupo de sanos es tan grande que su pequeño 5 % de falsos positivos arroja muchísimas más alarmas que el 99 % de aciertos sobre el minúsculo grupo de enfermos.



Lo más revelador es que **el mismo positivo significa cosas distintas según la prevalencia**. La prueba no cambia (sigue acertando el 99 % en enfermos y fallando el 5 % en sanos), pero su veredicto se reinterpreta según lo frecuente que sea la enfermedad en la población a la que se aplica. Si barremos la prevalencia de 0 a 50 % y recalculamos Bayes en cada punto, sale esta curva:

El mismo positivo cambia de significado con la prevalencia

La prueba no varía (sensibilidad 99 %, falsos positivos 5 %); cambia la población a la que se aplica.



IA para gente curiosa / Facsímil 12 / Capítulo 02 / 686f6c61

La curva sube deprisa al principio y luego se aplana: con la enfermedad rara (1 %) el positivo apenas alcanza el 16,7 %, no cruza el umbral del 50 % hasta una prevalencia cercana al 5 % y solo se acerca a la certeza cuando la enfermedad es muy común. Por eso una misma prueba sirve para confirmar en una población de riesgo y produce sobre todo falsas alarmas en un cribado masivo: el instrumento es el mismo, pero la tasa base lo recoloca. Es exactamente lo que un médico hace al decidir a quién conviene hacerle la prueba.

Naive Bayes: clasificar con una hipótesis “ingenua”

El teorema de Bayes con una sola evidencia ya es útil. Pero los problemas reales tienen muchas evidencias a la vez. Un correo no trae un solo indicio de spam: trae decenas de palabras, cada una un pequeño dato. Un ticket de soporte trae un texto entero. ¿Cómo aplicamos Bayes cuando la evidencia es un vector $E = (w_1, w_2, \dots, w_n)$ de muchos rasgos?

El teorema sigue valiendo, con la clase c como hipótesis:

$$P(c | w_1, \dots, w_n) = \frac{P(w_1, \dots, w_n | c) P(c)}{P(w_1, \dots, w_n)}$$

El problema está en la verosimilitud conjunta $P(w_1, \dots, w_n | c)$. Estimarla bien exigiría conocer cómo se combinan todas las palabras entre sí dentro de cada clase, una tabla astronómica que jamás tendríamos datos para llenar. Aquí entra la **hipótesis ingenua** (de ahí el nombre): suponer que, **dada la clase**, los rasgos son condicionalmente independientes entre sí. Es decir, que conocer la clase “spam” hace que la presencia de “gratis” no aporte información extra sobre la presencia de “premio”. Con esa suposición, la verosimilitud conjunta se descompone en un producto:

$$P(w_1, \dots, w_n \mid c) \approx \prod_{i=1}^n P(w_i \mid c)$$

En palabras: si suponemos que, una vez fijada la clase, las palabras no se influyen entre sí, la probabilidad de verlas todas juntas se vuelve el simple producto de la probabilidad de cada una por separado. Eso convierte una tabla imposible de llenar en una multiplicación de números que sí sabemos contar.

Y el clasificador elige la clase con mayor probabilidad posterior. Como el denominador $P(w_1, \dots, w_n)$ es el mismo para todas las clases, se puede ignorar para decidir, y basta comparar el numerador:

$$\hat{c} = \arg \max_c P(c) \prod_{i=1}^n P(w_i \mid c)$$

SÍMBOLO	SIGNIFICADO	EJEMPLO
$\hat{c}$	Clase elegida por el clasificador.	spam o legítimo.
$P(c)$	Probabilidad previa de cada clase.	Fracción de correos que son spam.
$P(w_i \mid c)$	Verosimilitud de cada palabra en la clase.	Frecuencia de “gratis” entre los spam.
$\arg \max_c$	La clase que maximiza el producto.	La hipótesis más creíble dadas las palabras.

En palabras: el clasificador prueba cada clase posible, multiplica lo frecuente que es esa clase por lo bien que sus palabras típicas explican el mensaje, y se queda con la clase que da el número mayor. No necesita la probabilidad exacta, solo saber cuál gana.

Un clasificador de spam, calculado a mano

Veámoslo con un corpus diminuto para que se vean todos los números. Tenemos seis mensajes de entrenamiento, tres spam y tres legítimos:

CLASE	MENSAJES DE ENTRENAMIENTO
spam	“gana dinero gratis” · “dinero gratis ya” · “gratis premio dinero”
legítimo	“reunión proyecto mañana” · “envío informe proyecto” · “reunión informe mañana”

El vocabulario completo tiene 10 palabras distintas: gana, dinero, gratis, ya, premio, reunión, proyecto, mañana, envío, informe. Cada clase aporta 9 palabras contadas con repetición. Los recuentos en la clase spam son: dinero 3, gratis 3, gana 1, ya 1, premio 1. Como las clases

tienen el mismo número de mensajes, las probabilidades previas son $P(\text{spam}) = P(\text{legítimo}) = 0,5$.

Para estimar $P(w | c)$ usamos el modelo multinomial con **suavizado de Laplace**: sumamos 1 a cada recuento para que ninguna palabra tenga probabilidad cero (si no, una sola palabra nunca vista en una clase anularía todo el producto). La fórmula es:

$$P(w | c) = \frac{\text{recuento}(w, c) + 1}{N_c + |V|}$$

donde $N_c = 9$ es el total de palabras de la clase y $|V| = 10$ el tamaño del vocabulario, así que el denominador es $9 + 10 = 19$.

SÍMBOLO	SIGNIFICADO	EJEMPLO
$\text{recuento}(w, c)$	Veces que la palabra w aparece en la clase c .	«gratis» sale 3 veces en spam.
N_c	Total de palabras de la clase c .	9 en cada clase del ejemplo.
$\$$	V	$\$$

En palabras: en lugar de dividir recuento entre total, sumamos un 1 a cada palabra antes de dividir. Así ninguna palabra recibe probabilidad cero, y una sola palabra nunca vista en una clase no puede tumbar todo el producto a cero. El precio es repartir un poquito de probabilidad a lo que aún no hemos visto.

Clasifiquemos el mensaje nuevo **“dinero gratis reunión”**. Necesitamos seis verosimilitudes:

PALABRA	EN SPAM	$P(W \text{SPAM})$	EN LEGÍTIMO	$P(W \text{LEGITIMO})$
dinero	3	$(3+1)/19 = 4/19$	0	$(0+1)/19 = 1/19$
gratis	3	$4/19$	0	$1/19$
reunión	0	$(0+1)/19 = 1/19$	2	$(2+1)/19 = 3/19$

Multiplicamos cada probabilidad previa por sus tres verosimilitudes:

$$P(\text{spam}) \prod P(w | \text{spam}) = 0,5 \cdot \frac{4}{19} \cdot \frac{4}{19} \cdot \frac{1}{19} = 0,5 \cdot \frac{16}{6859} = \frac{8}{6859}$$

$$P(\text{legítimo}) \prod P(w | \text{legítimo}) = 0,5 \cdot \frac{1}{19} \cdot \frac{1}{19} \cdot \frac{3}{19} = 0,5 \cdot \frac{3}{6859} = \frac{1,5}{6859}$$

Para convertir estos dos números en probabilidades que sumen 1, normalizamos dividiendo por su suma, 9,5/6859:

$$P(\text{spam} \mid \text{mensaje}) = \frac{8}{9,5} = 0,842, \quad P(\text{legítimo} \mid \text{mensaje}) = \frac{1,5}{9,5} = 0,158$$

El clasificador marca el mensaje como **spam con un 84,2 % de probabilidad**. Tiene sentido: dos de sus tres palabras son típicas de spam y solo una es neutra.

Por qué funciona pese a ser “ingenuo”

La hipótesis de independencia condicional es, casi siempre, **falsa**. En un correo de spam, “dinero” y “gratis” aparecen juntas mucho más de lo que predeciría tratarlas como independientes; no son sucesos que se ignoren entre sí. Entonces, ¿por qué Naive Bayes clasifica bien tan a menudo?

La clave es que para **decidir** una clase no necesitamos que las probabilidades sean exactas, solo que el orden entre clases sea correcto. Naive Bayes estima mal las magnitudes (sus probabilidades posterior suelen ser demasiado extremas, cercanas a 0 o a 1, justo el problema de calibración del Facsímil 7), pero acierta el ganador con frecuencia sorprendente.³ Es rápido, necesita pocos datos, se entrena en una sola pasada contando frecuencias y es difícil de batir como punto de partida (esa *baseline* del Facsímil 1, capítulo 11, que cualquier modelo más complejo debe superar para justificarse). Por eso sigue vivo en filtros de spam, clasificación de tickets y análisis de sentimiento, décadas después de inventarse.

Redes bayesianas: organizar muchas creencias

Naive Bayes hace una suposición drástica: que **todos** los rasgos son independientes dada la clase. Es eficaz, pero burdo. El mundo real tiene una estructura de dependencias más rica: unas variables influyen en otras de formas concretas, y muchas son independientes solo en ciertas condiciones. Necesitamos un lenguaje para representar exactamente esa estructura. Ese lenguaje son las **redes bayesianas**, formalizadas por Judea Pearl.⁴

Una red bayesiana es un **grafo dirigido acíclico** (un DAG, como los del capítulo de SAT y CSP, pero con otra lectura): cada nodo es una variable aleatoria, y cada flecha de X a Y expresa que X influye directamente en Y , que X es un “padre” de Y . El que sea acíclico importa: ninguna variable puede ser, dando la vuelta, causa de sí misma.

La idea central es que el grafo codifica una afirmación de independencia muy concreta: **cada variable es independiente de sus no-descendientes una vez conocidos sus padres**.

Dicho de otro modo, los padres de un nodo resumen toda la influencia que el resto de la red ejerce sobre él. Eso permite reescribir la regla de la cadena de forma mucho más barata. La conjunta general necesitaría condicionar cada variable sobre todas las anteriores; con la red, basta condicionar cada variable sobre sus padres:

$$P(X_1, \dots, X_n) = \prod_{i=1}^n P(X_i \mid \text{padres}(X_i))$$

SÍMBOLO	SIGNIFICADO	EJEMPLO
$P(X_1, \dots, X_n)$	Probabilidad conjunta de todas las variables.	Que coincidan robo, alarma y las dos llamadas.
$\prod_{i=1}^n$	Producto sobre todos los nodos de la red.	Una entrada de CPT por nodo.
$\text{padres}(X_i)$	Las variables con flecha directa hacia X_i .	Robo y Terremoto son los padres de Alarma.
$P(X_i \mid \text{padres}(X_i))$	La tabla (CPT) del nodo X_i .	Probabilidad de que suene la alarma según robo y terremoto.

En palabras: la probabilidad de un escenario completo es el producto de la probabilidad de cada variable mirando solo a sus padres, sin arrastrar el resto de la red. El grafo es justamente el mapa de qué se puede ignorar al calcular cada factor.

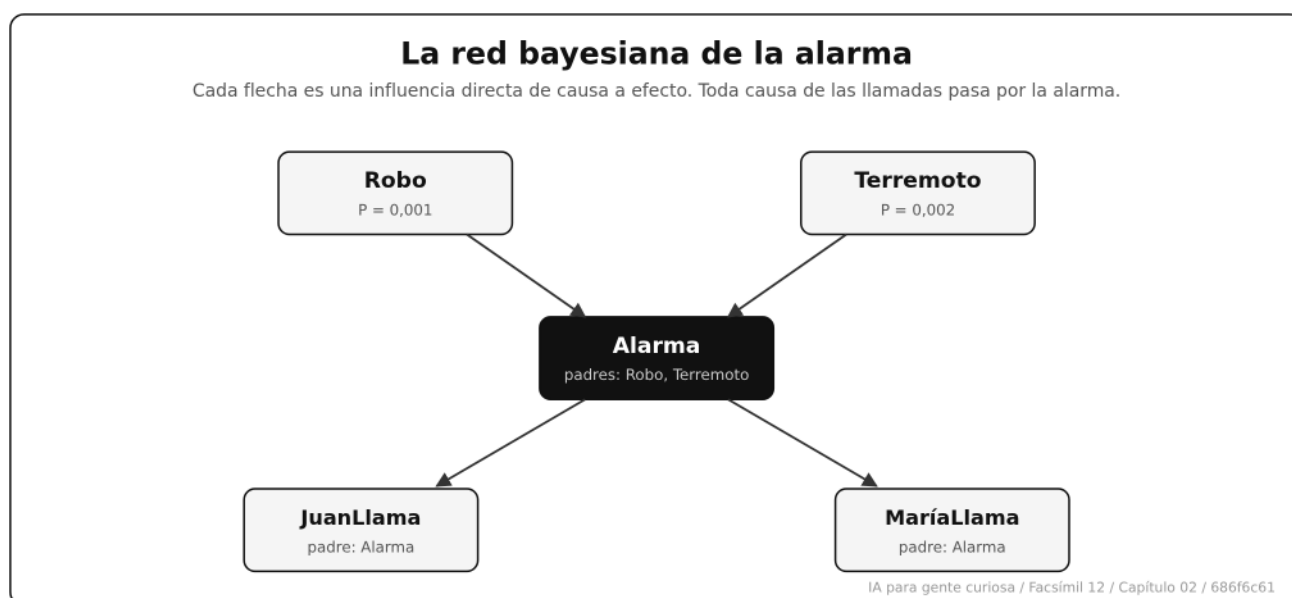
Esta **factorización** es la razón de ser de las redes bayesianas. La distribución conjunta de n variables booleanas tiene $2^n - 1$ números independientes, una cifra que explota enseguida. La red la sustituye por una tabla pequeña por nodo, la **tabla de probabilidad condicional** o CPT, que da la probabilidad de la variable para cada combinación de valores de sus padres. Si los padres son pocos, las tablas son pequeñas, y el producto reconstruye la conjunta entera sin necesidad de almacenarla.

El ejemplo canónico: la alarma de Pearl

El ejemplo de referencia, debido a Pearl, es una alarma antirrobo. Vives en una casa con una alarma que puede dispararse por un robo, pero también por un pequeño terremoto. Tienes dos vecinos, Juan y María, que te llaman al móvil si oyen la alarma, aunque no siempre: Juan a veces la confunde con el teléfono y llama sin motivo, y María a veces no la oye porque pone música alta. Cinco variables booleanas, cada una verdadera o falsa:

VARIABLE	SIGNIFICADO	PADRES
Robo	Hay un robo en la casa.	(ninguno)
Terremoto	Ha habido un terremoto.	(ninguno)
Alarma	La alarma se dispara.	Robo, Terremoto
JuanLlama	Juan te llama.	Alarma
MaríaLlama	María te llama.	Alarma

La estructura del grafo dice cosas precisas. Robo y Terremoto son causas independientes que confluyen en Alarma. Y, crucialmente, JuanLlama y MaríaLlama no dependen directamente del robo ni del terremoto: dependen solo de la alarma. Si supieras con certeza que la alarma sonó, saber además que hubo un robo no cambiaría tu predicción sobre si Juan llama. Toda la influencia del robo sobre Juan pasa, obligatoriamente, por la alarma. Ese es el contenido de las flechas:



Cada nodo lleva su CPT. Robo y Terremoto, al no tener padres, solo necesitan su probabilidad previa. La alarma necesita una fila por cada combinación de sus dos padres. Las llamadas, una fila por cada valor de la alarma. Estas son las tablas, las mismas del ejemplo original de Pearl:

CPT de Robo y de Terremoto (sin padres):

VARIABLE	$P(\text{VERDADERO})$
Robo	0,001
Terremoto	0,002

CPT de Alarma, condicionada a sus dos padres:

ROBO	TERREMOTO	$P(\text{ALARMA} = \text{SI})$
sí	sí	0,95
sí	no	0,94
no	sí	0,29
no	no	0,001

CPT de JuanLlama y MaríaLlama, condicionadas a la alarma:

ALARMA	$P(\text{JUAN}=\text{SI})$	$P(\text{MARIA}=\text{SI})$
sí	0,90	0,70
no	0,05	0,01

Con estas cuatro tablas, la red codifica la distribución conjunta completa de las cinco variables. Por ejemplo, la probabilidad de que no haya robo ni terremoto, suene la alarma y llamen los dos vecinos se calcula multiplicando una entrada de cada CPT:

$$\begin{aligned}
 P(\neg r, \neg t, a, j, m) &= P(\neg r) P(\neg t) P(a \mid \neg r, \neg t) P(j \mid a) P(m \mid a) \\
 &= 0,999 \cdot 0,998 \cdot 0,001 \cdot 0,90 \cdot 0,70 = 0,000628
 \end{aligned}$$

En palabras: para saber lo probable que es un escenario concreto, basta tomar de cada tabla la casilla que le corresponde —no robo, no terremoto, alarma sí, Juan sí, María sí— y multiplicarlas todas. Cada factor es una lectura directa de una CPT.

Es un evento poco probable (una falsa alarma que moviliza a los dos vecinos), y la red lo cuantifica sin esfuerzo. Treinta y un números harían falta para la conjunta a pelo; aquí han bastado las diez probabilidades de las CPT.

Cuándo dos variables son independientes: la d-separación

La estructura del grafo no solo factoriza la conjunta: permite **leer las independencias de un vistazo**, sin tocar un solo número, mediante un criterio gráfico llamado **d-separación**. La idea es seguir los caminos entre dos variables y ver si la información puede “fluir” por ellos o queda bloqueada. Hay tres patrones de conexión, y se comportan de forma distinta:

PATRÓN	FORMA	SIN OBSERVAR EL NODO CENTRAL	OBSERVANDO EL NODO CENTRAL
Cadena	$X \rightarrow Z \rightarrow Y$	la información fluye	el camino se bloquea
Bifurcación	$X \leftarrow Z \rightarrow Y$	la información fluye	el camino se bloquea
Colisión	$X \rightarrow Z \leftarrow Y$	el camino está bloqueado	el camino se abre

Las dos primeras son intuitivas: un padre común o un eslabón intermedio transmiten dependencia, y observarlo la corta. En nuestra red, JuanLlama y MaríaLlama están conectadas por la bifurcación $\text{Juan} \leftarrow \text{Alarma} \rightarrow \text{María}$. Sin saber nada de la alarma, las dos llamadas están correlacionadas: que Juan llame te hace creer más que la alarma sonó, y eso te hace esperar también la llamada de María. Pero **dada** la alarma, se vuelven independientes: si ya sabes que la alarma sonó, la llamada de Juan no añade nada sobre la de María. Eso es exactamente la independencia condicional que asumía Naive Bayes, aquí leída en el grafo.

El tercer patrón, la **colisión**, es el más sutil y el más interesante. Robo y Terremoto chocan en Alarma: $\text{Robo} \rightarrow \text{Alarma} \leftarrow \text{Terremoto}$. De partida, sin observar nada, son independientes; los robos no causan terremotos. Pero si **observas la alarma** (sabes que sonó), Robo y Terremoto pasan a estar correlacionados de forma negativa. Es el fenómeno llamado *explaining away*, descartar por explicación: si la alarma sonó y te enteras de que ha habido un terremoto, baja tu sospecha de robo, porque el terremoto ya explica la alarma. Una causa, al confirmarse, resta credibilidad a la otra. La lógica binaria no sabría representar esto; la red lo hace de forma natural.

Responder preguntas: inferencia por eliminación de variables

Tener la conjunta factorizada está muy bien, pero lo que queremos es **hacer preguntas**: dado que Juan y María han llamado, ¿qué probabilidad hay de que haya un robo? Esto es $P(\text{Robo} \mid \text{Juan}=\text{sí}, \text{María}=\text{sí})$. La consulta tiene una variable de interés (Robo), unas variables de evidencia (las dos llamadas, fijadas a “sí”) y unas variables ocultas que no aparecen ni en una ni en otra (Terremoto y Alarma) y que hay que marginalizar.

El método exacto es la **eliminación de variables**: usamos Bayes para pasar a la conjunta, sumamos sobre las ocultas y normalizamos. Por Bayes,

$$P(r \mid j, m) = \frac{P(r, j, m)}{P(j, m)} \propto P(r, j, m) = \sum_t \sum_a P(r) P(t) P(a \mid r, t) P(j \mid a) P(m \mid a)$$

SÍMBOLO	SIGNIFICADO	EJEMPLO
r	La variable que preguntamos (Robo), fijada a un valor.	¿Hubo robo, sí o no?
j, m	Las variables de evidencia, ya observadas.	Juan y María han llamado.
$\sum_t \sum_a$	Suma sobre las variables ocultas.	Recorrer terremoto y alarma, sí y no.
$\propto$	«Proporcional a»: falta dividir por $P(j, m)$.	Normalizar al final para que sume 1.

En palabras: para responder la consulta reconstruimos la conjunta, sumamos sobre las variables que no nos importan (terremoto y alarma) recorriendo todos sus valores, y al final normalizamos para que las dos opciones de robo vuelvan a sumar 1. El símbolo $\propto$ avisa de que, hasta ese último paso, manejamos números sin normalizar.

La clave para que esto sea barato es **empujar las sumas hacia dentro**: cada variable solo se suma sobre los factores que la contienen, no sobre todo el producto. Empecemos definiendo el factor que combina las dos llamadas en función de la alarma, $f(a) = P(j \mid a) P(m \mid a)$:

ALARMA	$P(J \mid A)$	$P(M \mid A)$	$F(A)$
sí	0,90	0,70	0,630
no	0,05	0,01	0,0005

Ahora, para cada combinación de Robo y Terremoto, sumamos sobre la alarma el producto $P(a \mid r, t) f(a)$. Llamemos a ese resultado $g(r, t)$:

$$g(r, t) = P(a=\text{sí} \mid r, t) \cdot 0,630 + P(a=\text{no} \mid r, t) \cdot 0,0005$$

ROBO	TERREMOTO	$P(A=SI \mid R, T)$	$G(R, T)$
sí	sí	0,95	$0,95 \cdot 0,630 + 0,05 \cdot 0,0005 = 0,598525$
sí	no	0,94	$0,94 \cdot 0,630 + 0,06 \cdot 0,0005 = 0,592230$
no	sí	0,29	$0,29 \cdot 0,630 + 0,71 \cdot 0,0005 = 0,183055$
no	no	0,001	$0,001 \cdot 0,630 + 0,999 \cdot 0,0005 = 0,001130$

A continuación sumamos sobre Terremoto, ponderando por su probabilidad previa ($P(t=sí) = 0,002$, $P(t=no) = 0,998$):

$$\text{Robo sí: } 0,002 \cdot 0,598525 + 0,998 \cdot 0,592230 = 0,592243$$

$$\text{Robo no: } 0,002 \cdot 0,183055 + 0,998 \cdot 0,001130 = 0,001493$$

Finalmente multiplicamos por la probabilidad previa de Robo ($P(r=sí) = 0,001$, $P(r=no) = 0,999$) para obtener los dos números sin normalizar:

$$P(r=sí, j, m) = 0,001 \cdot 0,592243 = 0,000592$$

$$P(r=no, j, m) = 0,999 \cdot 0,001493 = 0,001492$$

Normalizamos dividiendo por la suma, que es la evidencia $P(j, m) = 0,000592 + 0,001492 = 0,002084$:

$$P(\text{Robo} \mid \text{Juan, María}) = \frac{0,000592}{0,002084} = 0,284$$

El resultado es **28,4 %**. Vuelve a ser una lección sobre la tasa base: aunque las dos llamadas son indicios fuertes de robo, el robo era tan improbable de partida (una entre mil) que, incluso con ambos vecinos llamando, lo más probable sigue siendo que **no** haya robo. La evidencia ha multiplicado tu creencia por más de 280 (de 0,001 a 0,284), pero partía de tan abajo que no llega ni a la mitad. La red ha hecho automáticamente lo que a mano nos costaría: combinar muchas creencias coherentemente.

Cuando lo exacto no basta: inferencia aproximada

La eliminación de variables es exacta, pero su coste crece con la estructura de la red. En grafos grandes y densamente conectados, las sumas intermedias se vuelven intratables: el problema de inferencia exacta es, en general, NP-difícil, el mismo muro de complejidad que

vimos con SAT.⁵ Cuando eso ocurre, se renuncia a la respuesta exacta y se estima mediante **muestreo**: en lugar de sumar sobre todas las combinaciones, se simulan miles de mundos posibles según las CPT y se cuenta en qué fracción ocurre lo que preguntamos.

La forma más simple es el **muestreo directo**: recorrer la red de padres a hijos, sorteando cada variable según su CPT dado lo ya sorteado, hasta generar un “mundo” completo. Para responder a una consulta con evidencia se descartan los mundos que no encajan con lo observado (muestreo por rechazo) o se ponderan (muestreo por verosimilitud); técnicas más sofisticadas, como las cadenas de Markov de Montecarlo, exploran el espacio de forma más eficiente. La idea de fondo es la misma que reaparece, a otra escala, en los métodos modernos: cuando no puedes calcular una probabilidad, la aproximas con muchas simulaciones.⁶

En el día a día

Aunque parezcan formalismos de manual, estas ideas están funcionando constantemente a tu alrededor. El filtro de spam de tu correo es, en su versión clásica, un Naive Bayes contando palabras; los filtros modernos son más sofisticados, pero la intuición de “esta palabra hace más probable el spam” sigue ahí. Los sistemas de diagnóstico médico asistido combinan síntomas, pruebas y prevalencias exactamente con la mecánica del ejemplo de la enfermedad rara, y su mayor valor es precisamente recordar la tasa base que el instinto humano olvida.

Las redes bayesianas aparecen en diagnóstico de averías (un coche o una máquina industrial cuyos sensores apuntan a una causa raíz), en evaluación de riesgo crediticio, en sistemas de recomendación y en modelos de fiabilidad de equipos. Y cada vez que una aplicación combina varias señales débiles para tomar una decisión (un antifraude que junta ubicación, importe y hora; un detector que pondera varios indicios), por debajo hay una actualización de creencias que es Bayes, se llame así o no.

Hay además un sitio donde estas ideas viven sin que casi nadie lo mencione: dentro de los modelos de lenguaje. Un LLM es, en su núcleo, un modelo probabilístico que estima $P(\text{siguiente palabra} \mid \text{texto anterior})$. No razona con redes bayesianas explícitas, pero su salida es una distribución de probabilidad sobre el vocabulario, y todo lo que sabemos de probabilidad se aplica a ella.

Por qué debería importarte

Porque la incertidumbre no es un defecto que la IA vaya a eliminar; es la condición normal de cualquier sistema que actúa en el mundo real. La lógica del capítulo anterior te enseña a razonar cuando los hechos son seguros. Bayes te enseña a razonar cuando no lo son, que es casi siempre. Y te da algo que el instinto no tiene: un método para combinar la frecuencia de fondo de algo (la tasa base) con la fuerza de una evidencia concreta, sin sobreinterpretar ninguna de las dos.

Esto conecta directamente con cómo se debe usar un modelo de IA hoy. Cuando un LLM o un clasificador te da una probabilidad, esa cifra solo vale si está bien **calibrada**: si entre todos los casos en que dice “0,8” aproximadamente el 80 % resultan ciertos. Esa es la pregunta del Facsímil 7, capítulo 5, y es heredera directa de este: una probabilidad es una promesa sobre frecuencias, y Bayes es la gramática que la hace coherente. Entender el grado de creencia como un número con reglas es lo que te permite no tragarte una confianza inflada ni descartar una alerta solo porque “suena improbable”.

Dónde solía tropezar yo

ERROR	POR QUÉ ES UN ERROR	ANTÍDOTO
Olvidar la tasa base	Confundir $P(E H)$ (cómo de fiable es la prueba) con $P(H E)$ (la probabilidad real de la hipótesis) ignora lo raro que era la hipótesis previa.	Calcula siempre la evidencia $P(E)$ completa. Si la hipótesis es rara, hasta una prueba buena deja una probabilidad posterior baja.
Confundir $P(A B)$ con $P(B A)$	Son cantidades distintas; intercambiarlas es el error que produce el famoso 95 % falso en la prueba médica.	Escribe explícitamente qué condicionas sobre qué antes de poner números. La barra no es simétrica.
Tomar las probabilidades de Naive Bayes al pie de la letra	La hipótesis ingenua infla las cifras hacia 0 y 1; el orden suele ser correcto, pero la magnitud no es fiable.	Usa Naive Bayes para decidir la clase, no como probabilidad calibrada. Si necesitas la cifra, calíbrala.
Dibujar flechas como si fueran causalidad garantizada	En una red bayesiana la flecha codifica dependencia e influencia directa, pero la dirección la eliges tú; mal orientada, la factorización deja de ser económica.	Orienta las flechas de causa a efecto cuando puedas, y comprueba que cada nodo depende de pocos padres.
Creer que observar más siempre acerca dos variables	En una colisión ocurre lo contrario: observar el efecto común correlaciona causas antes independientes (<i>explaining away</i>).	Identifica el patrón (cadena, bifurcación, colisión) antes de afirmar una independencia.

Cómo encaja todo

Este capítulo es la bisagra del facsímil, y conviene verlo así: a un lado queda la certeza, al otro la creencia con grados. Venimos de la lógica del capítulo 1, donde cada hecho era verdadero o falso sin matices, y lo que hemos hecho aquí no ha sido tirarla a la basura, sino estirla. La verdad y la falsedad se han convertido en los dos extremos, 1 y 0, de una regla continua, y entre medias cabe todo lo que de verdad nos pasa por la cabeza cuando decidimos

con información incompleta. Por eso el capítulo siguiente, el 3 de este mismo Facsímil 12, viene a continuación de forma natural: explora qué hacer cuando ni siquiera podemos poner un número limpio a la creencia, cuando la incertidumbre no es probabilística sino imprecisa o vaga, y para eso recurre a los factores de certeza y a la lógica difusa. Si Bayes te exige una probabilidad previa exacta («el robo ocurre una vez entre mil»), el capítulo 3 acepta que muchas veces lo único honesto que puedes decir es «poco probable, pero no sabría cuantificarlo», y construye una herramienta para esa penumbra. Son dos respuestas distintas a la misma pregunta incómoda de cuánto fiarte.

Hacia atrás, el capítulo cierra un círculo muy concreto con el Facsímil 1. El Naive Bayes que aquí hemos calculado a mano es exactamente uno de los clasificadores clásicos del capítulo 11 de aquel facsímil, y no es casualidad que reaparezca: allí lo presentamos como la baseline, la línea de salida que cualquier modelo más complejo tiene que superar para justificar su coste. La razón de que sea tan buena referencia es justo lo que hemos visto: se entrena contando frecuencias en una sola pasada y casi nunca falla el orden de las clases. Y la lección de la tasa base reaparece allí con otro nombre, el de las clases desbalanceadas. Un detector de fraude que ve un caso de cada diez mil cae en el mismo espejismo que el positivo de la enfermedad rara: confundir lo fiable que es la señal con lo probable que es el suceso. La matriz de confusión y el coste de los errores de aquel capítulo son, vistos desde aquí, maneras prácticas de no dejarse engañar por una probabilidad previa minúscula.

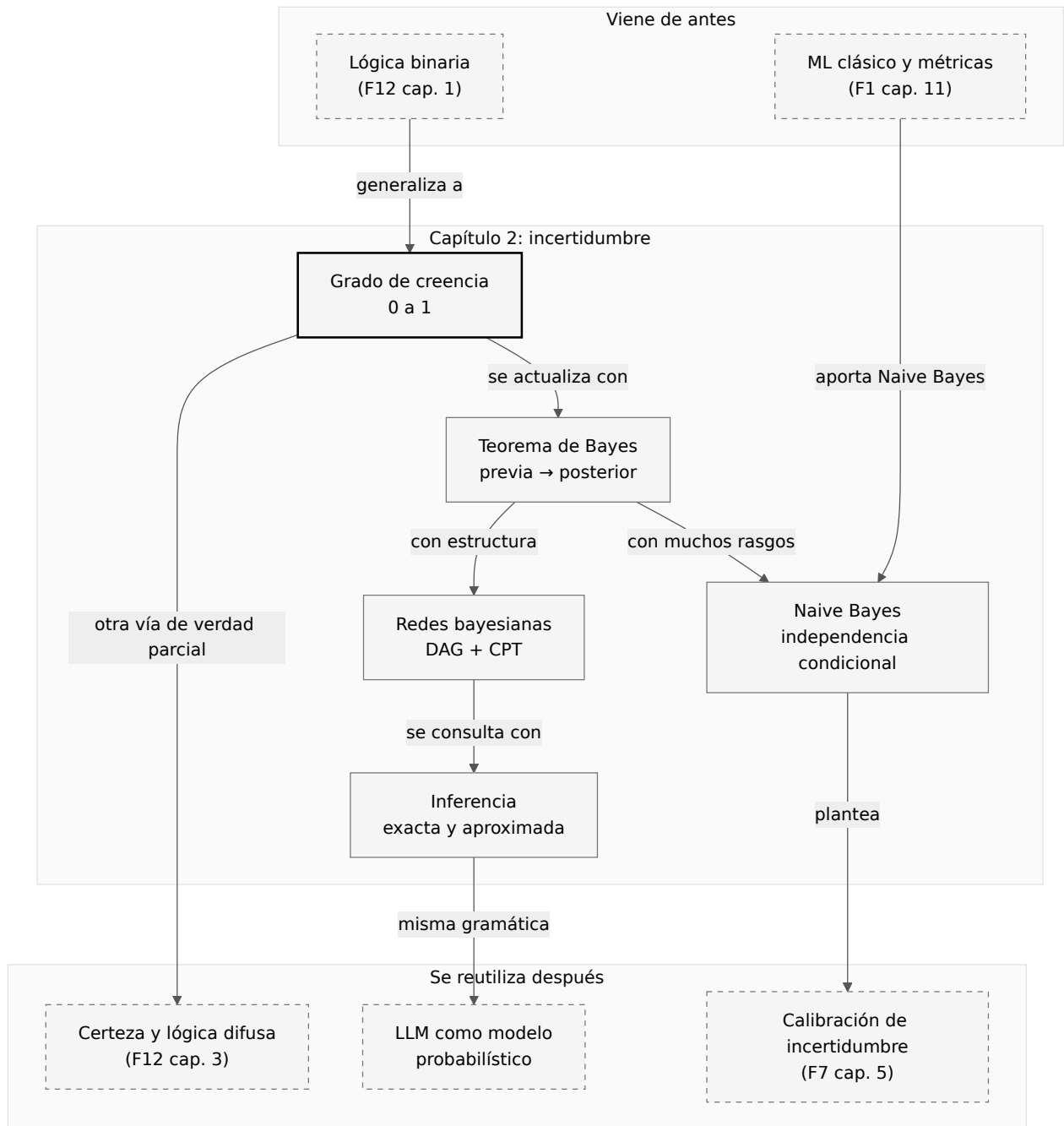
Hacia delante, el hilo más directo lleva al Facsímil 7, capítulo 5, sobre calibración e incertidumbre. Cuando un modelo —el Naive Bayes de aquí o un sistema mucho mayor— te dice «0,8», esa cifra es una promesa: que de cada cien veces que diga 0,8, alrededor de ochenta deberían acertar. Calibrar es comprobar si la promesa se cumple, y es la versión moderna de la pregunta que recorre todo este capítulo, «¿cuánto debería fiarme de esto?», pero formulada ya no para un cálculo a mano sino para un modelo que escupe miles de probabilidades al día. De hecho aquí ya avisamos del problema: Naive Bayes acierta el ganador pero miente en la magnitud, porque sus probabilidades se pegan a 0 y a 1. Esa patología tiene nombre y remedio, y son precisamente los del Facsímil 7. Es la diferencia entre un sistema que dice «99 % seguro» y se equivoca una de cada tres veces, y otro cuyo «80 %» puedes tomarte al pie de la letra.

Conecta también con lo que hoy llamamos IA moderna por una idea más sencilla de lo que parece. Un modelo de lenguaje, de los del Facsímil 3, no es en su núcleo más que una máquina probabilística: ante un texto, estima la probabilidad de cada posible palabra siguiente y elige según esa distribución. Cuando escribes «el cielo está» y el modelo continúa con «despejado», por dentro ha hecho lo mismo que nosotros con el spam: comparar verosimilitudes y quedarse con la más creíble dado el contexto. Toda la gramática de este capítulo —probabilidad condicional, marginalización, probabilidad posterior— se aplica a esa salida. Por eso entender Bayes no es un capricho de matemático: es lo que te permite leer con sentido crítico lo que afirma un chatbot, sabiendo que detrás hay una distribución de probabilidad, no una verdad revelada.

Y hay una conexión fácil de pasar por alto que el Facsímil 8 desarrolla: de dónde salen las probabilidades previas. Aquí te las hemos dado hechas —la enfermedad afecta al 1 %, el robo ocurre una vez entre mil—, pero en la práctica esos números se estiman a partir de datos, y

los datos arrastran sus propios sesgos. Si tu muestra sobrerrepresenta a un grupo, la probabilidad previa que aprendes queda torcida, y como Bayes la multiplica por la verosimilitud, ese sesgo se propaga sin resistencia hasta la probabilidad posterior. Un sistema de diagnóstico entrenado sobre una población distinta de la tuya parte de una tasa base equivocada y se equivoca con precisión matemática. Por eso el Facsímil 8, sobre datos y sesgos, es el complemento incómodo de este: Bayes es escrupulosamente honesto con las creencias que le metes, pero no las pone en duda; esa vigilancia te toca a ti.

Visto en conjunto, este capítulo no es una parada aislada sino un cruce de caminos: recoge el aprendizaje automático clásico que vino antes, presta su gramática a la calibración y a los modelos de lenguaje que vienen después, y deja una puerta abierta hacia las incertidumbres que ni siquiera se dejan escribir como una probabilidad. El diagrama lo resume:



Vocabulario aprendido

TÉRMINO	DEFINICIÓN
Probabilidad condicional	Probabilidad de un suceso dado que otro ha ocurrido, $P(A B)$. Condicionar reduce el universo a los casos donde B es cierto.
Teorema de Bayes	Regla que actualiza la creencia en una hipótesis al observar evidencia, invirtiendo la dirección de la condicional.
Probabilidad previa	Creencia en una hipótesis antes de ver la evidencia; incluye la tasa base.
Verosimilitud	Probabilidad de la evidencia suponiendo cierta la hipótesis, $P(E H)$.
Probabilidad posterior	Creencia en la hipótesis después de incorporar la evidencia; la probabilidad posterior de hoy es la previa de mañana.
Marginalización	Eliminar una variable de una conjunta sumando sobre todos sus valores.
Naive Bayes	Clasificador que aplica Bayes asumiendo independencia condicional entre rasgos dada la clase.
Independencia condicional	Dos variables son independientes una vez conocida una tercera.
Red bayesiana	Grafo dirigido acíclico que factoriza una distribución conjunta mediante CPT.
Tabla de probabilidad condicional (CPT)	Tabla que da la probabilidad de una variable para cada combinación de valores de sus padres.
d-separación	Criterio gráfico para decidir independencias condicionales según los patrones de cadena, bifurcación y colisión.
Eliminación de variables	Algoritmo de inferencia exacta que suma las variables ocultas para responder una consulta.

Antes de pasar página

- ☐ ¿Sé explicar por qué la lógica binaria del capítulo 1 no basta cuando los hechos son inciertos? (Si no, vuelve a «Entrando en el tema».)
- ☐ ¿Puedo escribir el teorema de Bayes y nombrar sus cuatro piezas? (Si no, vuelve a «El teorema de Bayes: dar la vuelta a la pregunta».)
- ☐ ¿Entiendo por qué un positivo en una prueba fiable puede dejar solo un 16,67 % de probabilidad de enfermedad? (Si no, vuelve a «El ejemplo que casi todo el mundo falla».)

- ☐ ¿Sé qué supone la hipótesis «ingenua» de Naive Bayes y por qué funciona aun siendo falsa? (Si no, vuelve a «Naive Bayes».)
- ☐ ¿Puedo escribir la factorización de la conjunta de una red bayesiana a partir de su grafo? (Si no, vuelve a «Redes bayesianas».)
- ☐ ¿Distingo cadena, bifurcación y colisión, y entiendo el *explaining away*? (Si no, vuelve a «Cuándo dos variables son independientes: la d-separación».)
- ☐ ¿He seguido el cálculo de $P(\text{Robo} \mid \text{Juan, María}) = 0,284$ por eliminación de variables? (Si no, vuelve a «Responder preguntas».)
- ☐ ¿He ejecutado el cuaderno y comprobado que `scikit-learn` y `pgmpy` reproducen los números?

En resumen

IDEA FUERZA	DETALLE
La probabilidad es lógica con grados.	El grado de creencia entre 0 y 1 generaliza el verdadero/falso, con reglas tan estrictas como las de la lógica.
Bayes invierte la pregunta.	Convierte “cómo de fiable es la prueba” en “qué probabilidad tengo de estar enfermo”, combinando la previa, la verosimilitud y evidencia.
La tasa base decide.	Ignorar lo raro que es una hipótesis previa es el error que produce el falso 95 %; la probabilidad posterior real era 16,67 %.
Naive Bayes acierta sin ser realista.	La independencia condicional es falsa, pero basta para ordenar bien las clases; por eso sigue siendo una baseline difícil de batir.
Las redes bayesianas organizan muchas creencias.	Un DAG con CPT factoriza la conjunta, hace legibles las independencias y permite inferir consultas exactas o aproximadas.
Es la gramática de lo moderno.	Un LLM estima probabilidades condicionales; calibrar y razonar sobre ellas exige las reglas de este capítulo.

Para saber más

Bayes, T. (1763). An essay towards solving a problem in the doctrine of chances. *Philosophical Transactions of the Royal Society of London*, 53, 370-418.
<https://doi.org/10.1098/rstl.1763.0053>

Duda, R. O., Hart, P. E. y Stork, D. G. (2001). *Pattern classification* (2.^a ed.). Wiley.

Koller, D. y Friedman, N. (2009). *Probabilistic graphical models: principles and techniques*. MIT Press.

Nilsson, N. J. (1986). Probabilistic logic. *Artificial Intelligence*, 28(1), 71-87.
[https://doi.org/10.1016/0004-3702\(86\)90031-7](https://doi.org/10.1016/0004-3702(86)90031-7)

Pearl, J. (1988). *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann.

Russell, S. y Norvig, P. (2021). *Artificial intelligence: a modern approach* (4.^a ed.). Pearson.

Notas

1. Nilsson, N. J. (1986). Probabilistic logic. *Artificial Intelligence*, 28(1), 71-87.
[https://doi.org/10.1016/0004-3702\(86\)90031-7](https://doi.org/10.1016/0004-3702(86)90031-7). Nilsson propuso una lógica probabilística que extiende la lógica clásica asignando probabilidades a las proposiciones en lugar de valores de verdad rígidos, y muestra que la lógica binaria es el caso límite cuando las probabilidades son 0 o 1. ↵
2. Bayes, T. (1763). An essay towards solving a problem in the doctrine of chances. *Philosophical Transactions of the Royal Society of London*, 53, 370-418.
<https://doi.org/10.1098/rstl.1763.0053>. El ensayo, publicado tras la muerte de Bayes por su amigo Richard Price, plantea por primera vez el problema de inferir la causa probable a partir de los efectos observados, que es la estructura de toda inferencia bayesiana posterior. ↵
3. Duda, R. O., Hart, P. E. y Stork, D. G. (2001). *Pattern classification* (2.^a ed.). Wiley. El texto analiza el clasificador bayesiano ingenuo y muestra por qué su frontera de decisión puede ser correcta aunque las estimaciones de densidad sean imprecisas, siempre que los errores no alteren cuál es la clase de mayor probabilidad. ↵
4. Pearl, J. (1988). *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann. Pearl introdujo las redes bayesianas como representación gráfica de la independencia condicional y desarrolló los algoritmos de propagación de creencias que hicieron tratable la inferencia probabilística en sistemas con muchas variables. ↵
5. Russell, S. y Norvig, P. (2021). *Artificial intelligence: a modern approach* (4.^a ed.). Pearson. Los capítulos sobre razonamiento probabilístico presentan la eliminación de variables y los métodos de Montecarlo, y demuestran que la inferencia exacta en redes bayesianas generales es computacionalmente difícil, lo que motiva los métodos aproximados. ↵
6. Koller, D. y Friedman, N. (2009). *Probabilistic graphical models: principles and techniques*. MIT Press. El libro es el tratado de referencia sobre modelos gráficos probabilísticos y cubre con detalle tanto la inferencia exacta como la familia de métodos de muestreo y de propagación aproximada para redes de gran tamaño. ↵

Capítulo 03: Cuando la verdad es difusa: factores de certeza y lógica difusa

Entrando en el tema

En el capítulo 1 de este facsímil la verdad era cómoda: una afirmación era verdadera o falsa, y punto. En el capítulo 2 dimos un paso y aceptamos que no siempre sabemos si algo es verdad; ahí la probabilidad nos dejó hablar de grados de creencia. Este capítulo trata un tercer problema, más sutil y muy fácil de confundir con el segundo: a veces el problema no es que no sepamos si algo es verdad, sino que la propia afirmación no tiene una frontera nítida.

«La habitación está calentita.» «El paciente tiene algo de fiebre.» «Es un hombre alto.» Ninguna de estas frases es verdadera o falsa de golpe. ¿A partir de qué temperatura una habitación deja de estar fresquita y empieza a estar calentita? ¿37,4 °C es «algo de fiebre» o todavía no? No hay un umbral mágico, y si lo pusiéramos a la fuerza tendríamos resultados absurdos: 37,49 °C sería «sano» y 37,50 °C sería «con fiebre», como si un cambio de una centésima cambiara el mundo.

Este es el terreno de la **lógica difusa**, propuesta por Lotfi Zadeh en 1965.¹ Su idea, casi de sentido común, fue radical para la época: dejemos que una afirmación sea verdadera *en cierto grado*, con un valor entre 0 y 1. No «tiene fiebre: sí o no», sino «tiene fiebre con grado 0,7». Antes de llegar ahí pasaremos por los **factores de certeza**, una técnica intermedia que los sistemas expertos clásicos usaron para manejar reglas que no eran del todo seguras. Las dos ideas conviven en el mismo territorio: el de un mundo que no se deja partir limpiamente en dos.

Dos cosas distintas que llamamos «incierto»

El error más común, y el que conviene desmontar antes de cualquier fórmula, es meter en el mismo saco dos fenómenos que no son lo mismo.

La **incertidumbre** es no saber. Lanzo una moneda y, antes de mirarla, no sé si salió cara o cruz. El hecho es perfectamente nítido (es cara o es cruz, sin medias tintas), pero mi conocimiento es incompleto. Eso lo modela la probabilidad, como vimos en el capítulo 2: $P(\text{cara}) = 0,5$ describe mi ignorancia, no una propiedad borrosa de la moneda.

La **imprecisión** o **vaguedad** es otra cosa. «Juan es alto.» Aquí no me falta información: puedo conocer su estatura al milímetro, 1,82 m, y aun así la frase no es simplemente verdadera o falsa, porque «alto» no tiene una frontera exacta. El problema no está en lo que sé, sino en el propio predicado, que admite grados.

ASPECTO	INCERTIDUMBRE (PROBABILIDAD)	IMPRECISIÓN (LÓGICA DIFUSA)
Qué desconozco	Si un hecho nítido ocurre o no.	Nada: el hecho es vago por naturaleza.
Ejemplo	«¿Lloverá mañana?»	«Hoy hace calor.»
Qué mide el número	Grado de creencia en que algo es cierto.	Grado en que algo pertenece a una categoría.
Al observar la realidad	La duda se resuelve: llovió o no llovió.	La vaguedad permanece: 24 °C sigue siendo «calorcito» a medias.
Suma sobre alternativas	Las probabilidades de un suceso suman 1.	Los grados de pertenencia no tienen por qué sumar 1.

Ese último punto de la tabla es el más revelador y conviene grabárselo. Si hoy estamos a 22 °C, la temperatura puede pertenecer a «templado» con grado 0,8 y a «caliente» con grado 0,1 a la vez, y esos números **no suman 1** ni tienen por qué. No estoy repartiendo una probabilidad entre categorías excluyentes; estoy diciendo cuánto encaja un mismo valor en cada etiqueta solapada. Russell y Norvig insisten en esta distinción precisamente porque mezclarla lleva a usar la herramienta equivocada: la teoría de conjuntos difusos trata el grado de verdad de proposiciones vagas, no la probabilidad de proposiciones inciertas.²

La lección práctica: si tu pregunta es «¿pasará esto?», probablemente quieres probabilidad. Si tu pregunta es «¿cuánto se cumple esto?», probablemente quieres lógica difusa.

Factores de certeza: el apañío de los sistemas expertos

Antes de Zadeh llegara a las máquinas de control, hubo un problema más terrenal. En los años setenta se construyeron los primeros **sistemas expertos**: programas que codificaban el conocimiento de un especialista como un montón de reglas SI-ENTONCES. El más famoso fue MYCIN, desarrollado en Stanford para recomendar tratamientos de infecciones bacterianas.³ Y MYCIN se topó de frente con la vaguedad del conocimiento médico real: un médico no dice «si hay fiebre alta, entonces es gripe» como una verdad absoluta. Dice algo más parecido a «si hay fiebre alta, *tiendo a pensar* que es gripe, pero no estoy seguro del todo».

Los autores de MYCIN no querían meterse en la probabilidad bayesiana completa (faltaban datos para estimar tantas probabilidades, y el capítulo 2 ya nos avisó de lo caras que salen las tablas conjuntas). Inventaron un atajo: el **factor de certeza** o CF (del inglés *certainty factor*), un número entre -1 y +1 que resume cuánto refuerza o debilita una evidencia a una hipótesis.⁴

VALOR DEL CF	LECTURA
$CF = +1$	Certeza total de que la hipótesis es cierta.
CF entre 0 y 1	La evidencia apoya la hipótesis, con más fuerza cuanto más cerca de 1.
$CF = 0$	La evidencia no dice nada a favor ni en contra.
CF entre -1 y 0	La evidencia va en contra de la hipótesis.
$CF = -1$	Certeza total de que la hipótesis es falsa.

Cada regla lleva su propio CF, que mide cuánto creería el experto en la conclusión si la premisa fuera totalmente cierta:

$$\text{SI } e \text{ ENTONCES } h \quad (CF_{\text{regla}})$$

En palabras: si se observa la evidencia e , se apuesta por la hipótesis h con una fuerza CF_{regla} ; ese número, que fija el experto, dice cuánto creería en h si e fuese completamente segura.

Pero la premisa no siempre es totalmente cierta. Si solo estamos seguros de la evidencia e con un grado $CF(e)$, la conclusión se **atenúa**. MYCIN propaga así, dejando pasar solo la parte positiva de la evidencia:

$$CF(h, e) = CF_{\text{regla}} \cdot \max(0, CF(e))$$

En palabras: la certeza que una regla aporta a su hipótesis es la confianza de la regla multiplicada por la certeza de la evidencia, pero solo cuando esa evidencia juega a favor; si la evidencia es dudosa o va en contra, la parte positiva vale cero y la regla no aporta nada.

SÍMBOLO	SIGNIFICADO	EJEMPLO
e	La evidencia observada, premisa de la regla.	«el paciente tiene fiebre alta».
h	La hipótesis que la regla apoya, su conclusión.	«el paciente tiene gripe».
CF_{regla}	Confianza en la regla si la premisa fuese segura.	0,8 para «fiebre alta → gripe».
$CF(e)$	Certeza que tenemos sobre la evidencia.	0,5 si la fiebre solo es probable.
$CF(h, e)$	Certeza que la regla aporta a la hipótesis.	$0,8 \cdot 0,5 = 0,4$.

Combinar evidencias: cuando dos reglas opinan a la vez

La parte interesante llega cuando **dos reglas distintas** apoyan la misma hipótesis. Imagina dos pistas independientes que sugieren gripe: la fiebre y el dolor muscular. Cada una aporta su CF, y necesitamos fundirlos en una sola creencia. MYCIN usa una regla de combinación que depende de los signos:

$$CF(x, y) = \begin{cases} x + y(1 - x) & \text{si } x, y \geq 0 \\ [4pt] x + y(1 + x) & \text{si } x, y \leq 0 \\ [4pt] \frac{x + y}{1 - \min(|x|, |y|)} & \text{si signos opuestos} \end{cases}$$

En palabras: para fundir dos creencias sobre la misma hipótesis, si las dos empujan en el mismo sentido se suman descontando lo que ya cubría la otra, de modo que nunca pasan de 1 ni bajan de -1 ; si empujan en sentidos opuestos, se restan y se normalizan dividiendo por lo que tienen en común.

SÍMBOLO	SIGNIFICADO	EJEMPLO
x	Creencia ya acumulada en la hipótesis.	0,8 tras la primera regla.
y	Creencia que aporta la nueva evidencia.	0,6 de la segunda regla.
$CF(x, y)$	Creencia combinada de ambas.	0,92 al unir las.

CASO	IDEA INTUITIVA
Ambos positivos	Dos pistas a favor refuerzan, pero con rendimientos decrecientes: nunca pasan de 1.
Ambos negativos	Dos pistas en contra refuerzan la negación, sin bajar de -1 .
Signos opuestos	Una pista a favor y otra en contra se cancelan parcialmente.

Hagamos un ejemplo numérico completo. El paciente tiene fiebre alta (segura, $CF = 1$) y dolor muscular (seguro, $CF = 1$). Hay dos reglas:

- R1: SI fiebre alta ENTONCES gripe, con $CF_{\text{regla}} = 0,8$.
- R2: SI dolor muscular ENTONCES gripe, con $CF_{\text{regla}} = 0,6$.

Cada regla aporta su CF a la hipótesis «gripe»:

$$CF_1 = 0,8 \cdot 1 = 0,8, \quad CF_2 = 0,6 \cdot 1 = 0,6$$

Como ambos son positivos, los combinamos con la primera rama:

$$CF = 0,8 + 0,6(1 - 0,8) = 0,8 + 0,6 \cdot 0,2 = 0,8 + 0,12 = 0,92$$

Dos pistas a favor han subido la creencia de 0,8 a 0,92, pero ojo: no a 1,4. La fórmula respeta el techo. Ahora supongamos que aparece una tercera regla en contra, porque el paciente no tiene tos, lo cual es raro en una gripe:

- R3: SI no hay tos ENTONCES gripe, con $CF_{\text{regla}} = -0,4$.

Combinamos el 0,92 acumulado con el $-0,4$. Signos opuestos, tercera rama:

$$CF = \frac{0,92 + (-0,4)}{1 - \min(0,92, 0,4)} = \frac{0,52}{1 - 0,4} = \frac{0,52}{0,6} \approx 0,87$$

La evidencia en contra ha rebajado la creencia de 0,92 a 0,87, pero sin tumbarla. Para las premisas compuestas, MYCIN usa los mismos operadores que veremos en la parte difusa: el **mínimo** para la conjunción (Y) y el **máximo** para la disyunción (O).

$$CF(e_1 \wedge e_2) = \min(CF(e_1), CF(e_2)), \quad CF(e_1 \vee e_2) = \max(CF(e_1), CF(e_2))$$

En palabras: cuando una premisa une varias condiciones, la certeza de un «Y» es la de la condición más floja (su eslabón más débil) y la de un «O» es la de la condición más fuerte.

SÍMBOLO	SIGNIFICADO	EJEMPLO
e_1, e_2	Las condiciones que forman la premisa.	«fiebre alta» y «dolor muscular».
$\min(\cdot)$	Mínimo de los grados, para el «Y».	$\min(0,5, 0,9) = 0,5$.
$\max(\cdot)$	Máximo de los grados, para el «O».	$\max(0,5, 0,9) = 0,9$.

Por qué se usaron y por qué se quedaron cortos

Los factores de certeza funcionaron en la práctica y eran fáciles de explicar a un médico: cada regla tenía su número, y el sistema podía mostrar la cadena de reglas que llevó a una conclusión. Eso conecta de lleno con el capítulo 4, donde los sistemas expertos se apoyan en esta misma trazabilidad para justificar una decisión. Pero tenían un defecto de fondo que sus propios autores reconocieron: los CF no son probabilidades, aunque a veces se comporten parecido. La fórmula de combinación supone que las evidencias son independientes y modulares, algo que casi nunca se cumple, y puede dar resultados inconsistentes cuando las reglas se encadenan mucho. Con el tiempo, las **redes bayesianas** del capítulo 2 ofrecieron un fundamento más sólido para la incertidumbre, y los factores de certeza quedaron como una pieza histórica. Pero su espíritu (reglas graduales, fáciles de auditar) sigue vivo cada vez que un sistema asigna una puntuación de confianza a una conclusión.

Conjuntos difusos: pertenecer a medias

Volvamos a la vaguedad pura, que es donde Zadeh hizo su aportación. En la teoría de conjuntos de toda la vida, un elemento pertenece o no pertenece a un conjunto. El conjunto de los «mayores de edad» es nítido: con 17 años y 364 días estás fuera; con 18, dentro. La función que describe esa pertenencia solo toma dos valores, 0 o 1.

Un **conjunto difuso** rompe esa frontera. Su **función de pertenencia** ya no devuelve 0 o 1, sino cualquier valor del intervalo $[0, 1]$:

$$\mu_A : X \rightarrow [0, 1]$$

En palabras: una función de pertenencia toma cualquier valor del universo y le asigna un número entre 0 y 1 que dice cuánto encaja ese valor en el conjunto difuso, desde nada en absoluto (0) hasta del todo (1).

SÍMBOLO	SIGNIFICADO	EJEMPLO
A	El conjunto difuso.	«Temperatura caliente».
X	El universo de discurso: los valores posibles.	Grados centígrados de 0 a 40.
x	Un valor concreto del universo.	$x = 22\text{ }^{\circ}\text{C}$.
$\mu_A(x)$	Grado de pertenencia de x al conjunto A .	$\mu_{\text{caliente}}(22) = 0,1$.

Si $\mu_A(x) = 1$, el valor pertenece de lleno; si $\mu_A(x) = 0$, está completamente fuera; y todo lo de en medio es pertenencia parcial. Una temperatura de $22\text{ }^{\circ}\text{C}$ puede pertenecer a «templado» con grado 0,8 y a «caliente» con grado 0,1: encaja mucho en una etiqueta y un poco en la vecina.

Las formas habituales de una función de pertenencia

En la práctica no se dibujan curvas caprichosas. Se usan unas pocas formas sencillas, porque son fáciles de definir con dos o cuatro números y rápidas de calcular. Las dos más comunes son la **triangular** y la **trapezoidal**.

La triangular sube desde 0 hasta un pico y vuelve a bajar. Se define con tres puntos: dónde empieza a subir (a), dónde está el pico (b) y dónde acaba de bajar (c):

$$\mu_{\Delta}(x; a, b, c) = \max\left(\min\left(\frac{x - a}{b - a}, \frac{c - x}{c - b}\right), 0\right)$$

En palabras: el grado sube en línea recta desde 0 en el punto a hasta 1 en el pico b , y baja igual de recto hasta 0 en c ; fuera del intervalo $[a, c]$ vale 0.

SÍMBOLO	SIGNIFICADO	EJEMPLO
x	Valor de entrada que evaluamos.	22 °C.
a, b, c	Pie de subida, pico y pie de bajada.	10, 20, 30 para «templado».
$\mu_{\Delta}(x)$	Grado de pertenencia resultante.	0,8 a 22 °C.

La trapezoidal es igual pero con una meseta plana arriba, útil cuando hay un rango de valores que pertenecen «del todo». Se define con cuatro puntos:

$$\mu_{\text{trap}}(x; a, b, c, d) = \max\left(\min\left(\frac{x - a}{b - a}, 1, \frac{d - x}{d - c}\right), 0\right)$$

En palabras: funciona como la triangular, pero con una meseta plana en 1 entre b y c : se sube hasta b , todos los valores entre b y c pertenecen del todo, se baja desde c , y fuera de $[a, d]$ el grado es 0.

SÍMBOL O	SIGNIFICADO	EJEMPLO
x	Valor de entrada que evaluamos.	23 °C.
a, b, c, d	Pie izquierdo, inicio y fin de la meseta, pie derecho.	18, 20, 24, 26 para «temperatura confortable».
$\mu_{\text{trap}}(x)$	Grado de pertenencia resultante.	1 en toda la meseta.

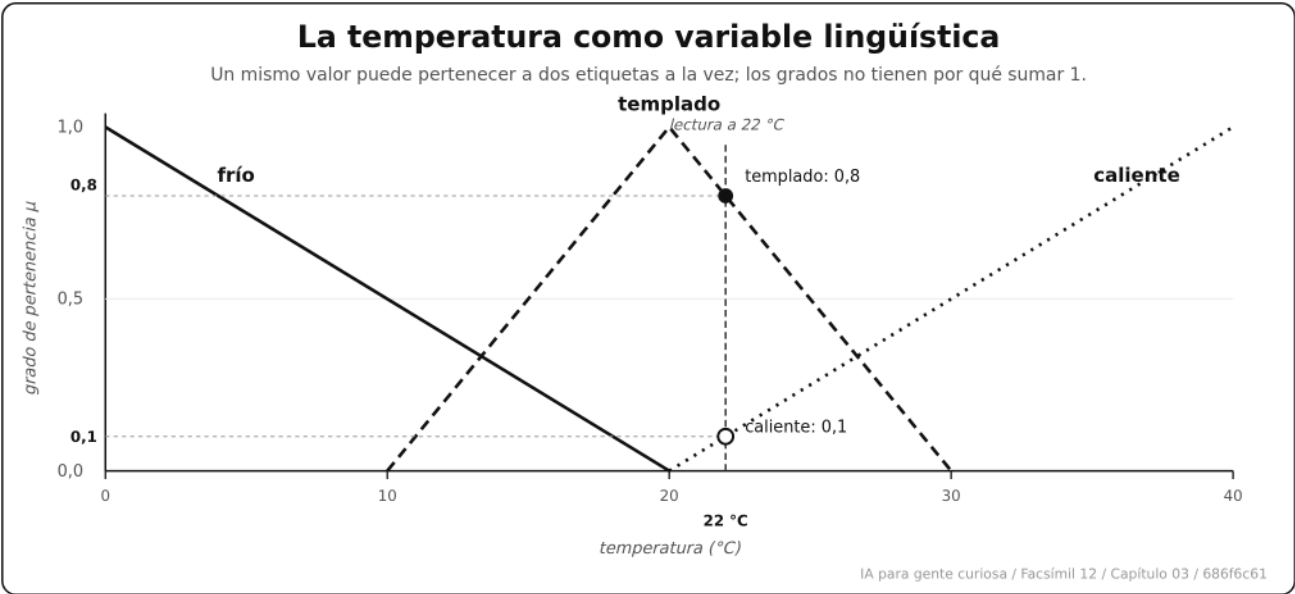
FORMA	PARÁMETROS	CUÁNDO CONVIENE
Triangular	a, b, c	Un valor «ideal» único y descenso a ambos lados.
Trapezoidal	a, b, c, d	Un rango entero que pertenece de lleno (la meseta).

Variables lingüísticas: poner palabras a los números

Aquí aparece una de las ideas más bonitas de Zadeh: la **variable lingüística**. En vez de tratar la temperatura como un número, la tratamos como una variable cuyos valores son *palabras*: frío, templado, caliente. Cada palabra es una **etiqueta** asociada a un conjunto

difuso con su función de pertenencia. Así, la frase «hace calor» deja de ser metáfora y pasa a tener una traducción matemática precisa.

El siguiente diagrama muestra tres etiquetas solapadas sobre el eje de temperatura. Fíjate en cómo se montan unas sobre otras: ese solapamiento es justo lo que permite que un valor pertenezca a dos categorías a la vez.



Con estas tres etiquetas, definidas como triángulos, la lectura de 22 °C es directa. La etiqueta «frío» tiene pico en 0 °C y llega a cero en 20 °C, así que a 22 °C ya vale 0. «Templado» es un triángulo con pico en 20 °C que baja hasta 0 en 10 y en 30; a 22 °C su grado es $(30 - 22)/10 = 0,8$. «Caliente» sube desde 0 en 20 °C hasta 1 en 40 °C; a 22 °C vale $(22 - 20)/20 = 0,1$. Esos son los dos puntos marcados en el diagrama.

ETIQUETA	DEFINICIÓN TRIANGULAR	M A 22 °C
frío	$\triangle(x; -, 0, 20)$	0
templado	$\triangle(x; 10, 20, 30)$	0,8
caliente	$\triangle(x; 20, 40, -)$	0,1

Operadores y reglas difusas

Una vez que tenemos grados de verdad, necesitamos combinarlos. La lógica difusa hereda los tres conectores de la lógica clásica, pero los reescribe para que funcionen con valores continuos. Las definiciones estándar de Zadeh son:

$$\mu_{A \cap B}(x) = \min(\mu_A(x), \mu_B(x)) \quad (\text{conjunción, Y})$$

En palabras: algo cumple «A y B» tanto como cumpla la condición que peor lleva, así que se toma el menor de los dos grados.

$$\mu_{A \cup B}(x) = \max(\mu_A(x), \mu_B(x)) \quad (\text{disyunción, O})$$

En palabras: algo cumple «A o B» tanto como cumpla la condición que mejor lleva, así que se toma el mayor de los dos grados.

$$\mu_{\neg A}(x) = 1 - \mu_A(x) \quad (\text{complemento, NO})$$

En palabras: el grado de «no A» es lo que le falta a A para llegar a 1; si algo pertenece a «caliente» con 0,7, pertenece a «no caliente» con 0,3.

SÍMBOLO	SIGNIFICADO	EJEMPLO
$\mu_A(x)$	Grado de x en el conjunto A .	templado a 22 °C: 0,8.
$\mu_B(x)$	Grado de x en el conjunto B .	caliente a 22 °C: 0,1.
min, max, 1–	Operadores para Y, O y NO.	$\min(0,8, 0,1) = 0,1$.

OPERADOR	REGLA DIFUSA	COMPROBACIÓN CON EXTREMOS
Y (conjunción)	mínimo de los grados	$\min(1, 1) = 1$, $\min(1, 0) = 0$: coincide con el AND clásico.
O (disyunción)	máximo de los grados	$\max(0, 0) = 0$, $\max(1, 0) = 1$: coincide con el OR clásico.
NO (complemento)	uno menos el grado	$1 - 1 = 0$, $1 - 0 = 1$: coincide con la negación clásica.

Lo elegante es que, si solo metemos ceros y unos, estos operadores se comportan exactamente como los booleanos de toda la vida. La lógica difusa **contiene** a la clásica como caso particular, igual que en el capítulo 5 del Facsímil 2 vimos que un CSP contiene a SAT. Ross subraya este punto: los conectores difusos generalizan los clásicos sin contradecirlos, lo que permite construir reglas que degradan suavemente en lugar de saltar de golpe.⁵

Con los operadores en la mano, una **regla difusa SI-ENTONCES** tiene la misma pinta que una regla normal, pero sus términos son etiquetas difusas:

SI servicio es *excelente* O comida es *buena* ENTONCES propina es *alta*

En palabras: si el servicio es excelente o la comida es buena, la propina debería ser alta; pero como cada etiqueta se cumple solo en cierto grado, la conclusión «propina alta» se activará en ese mismo grado, no de golpe.

La diferencia es que la premisa no es verdadera o falsa: se cumple en cierto grado, y ese grado «moja» a la conclusión. Veámoslo de principio a fin.

Inferencia de Mamdani: un controlador de propina paso a paso

El método más conocido para razonar con reglas difusas es la **inferencia de Mamdani**, propuesta por Ebrahim Mamdani y Sedrak Assilian en 1975 para controlar una máquina de vapor cuando las ecuaciones clásicas se quedaban cortas.⁶ Tiene cuatro pasos: fuzzificar, evaluar las reglas, agregar y defuzzificar. Vamos a recorrerlos enteros con números, usando el ejemplo clásico de calcular la propina de un restaurante en función de la calidad del servicio y de la comida.

El problema. Dos entradas, ambas valoradas de 0 a 10: la calidad del *servicio* y la calidad de la *comida*. Una salida: el *porcentaje de propina*, de 0 a 25 %. Definimos las etiquetas así:

VARIABLE	ETIQUETAS (CONJUNTOS DIFUSOS)
servicio (0-10)	malo $\triangle(-, 0, 5)$, aceptable $\triangle(0, 5, 10)$, excelente $\triangle(5, 10, -)$
comida (0-10)	mala $\triangle(-, 0, 5)$, buena $\triangle(5, 10, -)$
propina (0-25 %)	baja $\triangle(-, 0, 12,5)$, media $\triangle(0, 12,5, 25)$, alta $\triangle(12,5, 25, -)$

Y tres reglas, que resumen el sentido común de cualquier comensal:

- R1: SI servicio es *malo* O comida es *mala* ENTONCES propina es *baja*.
- R2: SI servicio es *aceptable* ENTONCES propina es *media*.
- R3: SI servicio es *excelente* O comida es *buena* ENTONCES propina es *alta*.

Supongamos una cena concreta: el servicio fue notable, *servicio* = 7, y la comida estuvo muy bien, *comida* = 8.

Antes de meternos en los cálculos, este es el recorrido completo de un extremo a otro: dos números crudos entran por la izquierda y salen, por la derecha, convertidos en una sola propina. Cada paso transforma un poco más la información hasta colapsarla en una cifra.

La inferencia de Mamdani de principio a fin

De dos números crudos a una propina concreta, en cuatro pasos encadenados.

1 · Fuzzificar

números → grados

servicio 7:

aceptable 0,6
excelente 0,4

comida 8:

buena 0,6

grados

2 · Evaluar reglas

$Y = \text{mín} \cdot O = \text{máx}$

R1 (O): $\text{máx}(0;0) = 0$

R2: aceptable = 0,6

R3 (O): $\text{máx}(0,4;0,6)$
= 0,6

cada fuerza recorta

conjuntos

3 · Agregar

máximo punto a punto



media + alta, techo 0,6

una figura

4 · Defuzzificar

centroide de la figura

$z^* = \int z \cdot \mu \, dz / \int \mu \, dz$

centro de gravedad

≈ 14,2 %

la propina final

IA para gente curiosa / Facsímil 12 / Capítulo 03 / 686f6c61

Paso 1: fuzzificación

Fuzzificar es traducir los números crudos a grados de pertenencia. Para *servicio* = 7:

ETIQUETA DE SERVICIO	CÁLCULO	GRADO
malo	$7 > 5$	0
aceptable	$(10 - 7)/5$	0,6
excelente	$(7 - 5)/5$	0,4

Para *comida* = 8:

ETIQUETA DE COMIDA	CÁLCULO	GRADO
mala	$8 > 5$	0
buena	$(8 - 5)/5$	0,6

Paso 2: evaluación de las reglas

Cada regla calcula la fuerza de su premisa con los operadores difusos, y esa fuerza limita su conclusión. La premisa con «O» usa el máximo; con «Y» usaría el mínimo.

REGLA	PREMISA	OPERACIÓN	FUERZA DE ACTIVACIÓN
R1	servicio malo (0) O comida mala (0)	$\max(0, 0)$	0
R2	servicio aceptable (0,6)	—	0,6
R3	servicio excelente (0,4) O comida buena (0,6)	$\max(0,4, 0,6)$	0,6

R1 no se activa en absoluto (fuerza 0), así que «propina baja» no contribuye. R2 activa «propina media» con fuerza 0,6 y R3 activa «propina alta» con fuerza 0,6. En Mamdani, esa fuerza **recorta** (operación mínimo) la función de pertenencia de la conclusión: el triángulo de «media» se queda con un techo plano a la altura 0,6, y lo mismo le pasa a «alta».

Paso 3: agregación

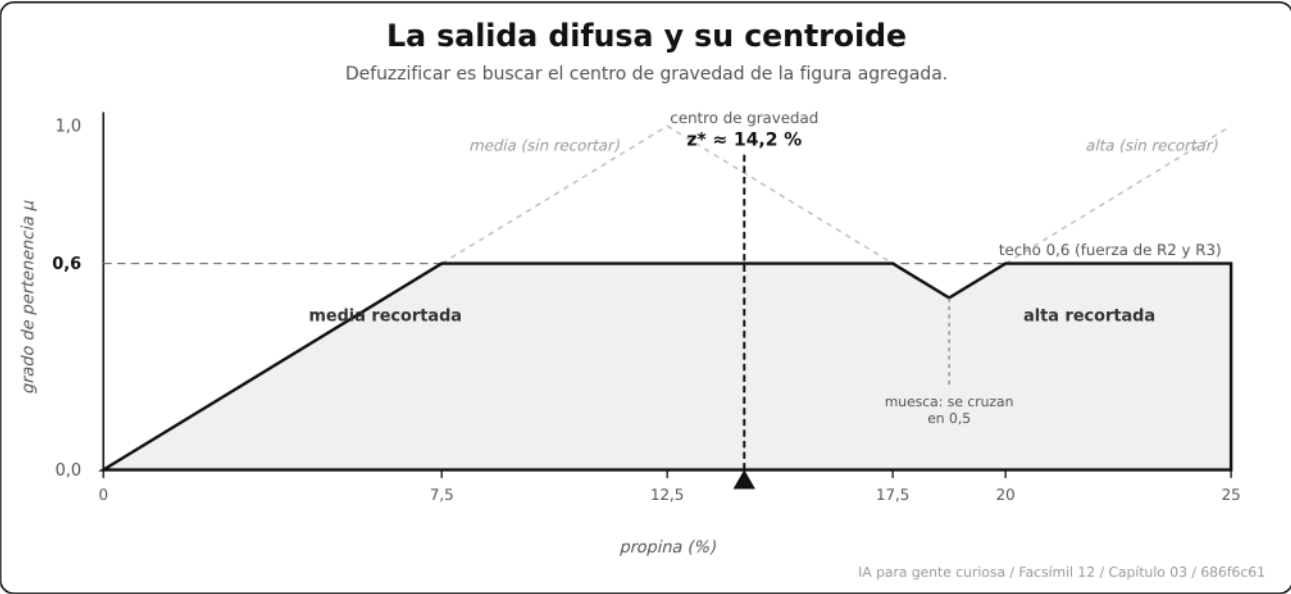
Ahora superponemos todas las conclusiones recortadas en una sola figura, tomando el **máximo** punto a punto. El resultado es un único conjunto difuso de salida, con esta forma sobre el eje de la propina:

$$\mu(z) = \max \left(\min(0,6, \mu_{\text{media}}(z)), \min(0,6, \mu_{\text{alta}}(z)) \right)$$

En palabras: en cada punto del eje de la propina ponemos la mayor de las alturas que aportan las reglas activas, después de recortar cada conclusión a su fuerza; es decir, superponemos «media» y «alta» recortadas y nos quedamos con el contorno más alto.

SÍMBOLO	SIGNIFICADO	EJEMPLO
z	Un valor del eje de salida, la propina.	15 %.
$\mu_{\text{media}}(z), \mu_{\text{alta}}(z)$	Pertenencia de z a cada conclusión.	faldas de «media» y «alta».
0,6	Fuerza que recorta cada regla (su techo).	de R2 y R3.
$\mu(z)$	Altura de la figura agregada en z .	el contorno superior.

El siguiente diagrama dibuja esa figura agregada. Sube por la izquierda (la falda de «media»), se aplan a 0,6, tiene una pequeña muesca donde «media» y «alta» se cruzan por debajo del techo, y vuelve a 0,6 hasta el extremo derecho (la cima recortada de «alta»).



Paso 4: defuzzificación por centroide

La figura agregada todavía es difusa, pero el camarero necesita un número: ¿cuánto dejo de propina? **Defuzzificar** es colapsar esa figura en un valor concreto. El método más usado es el **centroide** o centro de gravedad: el punto del eje donde la figura «se equilibraría» si fuera una placa con peso.

$$z^* = \frac{\int z \mu(z) dz}{\int \mu(z) dz} \approx \frac{\sum_i z_i \mu(z_i)}{\sum_i \mu(z_i)}$$

En palabras: el valor final es el promedio de todas las posiciones del eje, pero ponderando cada una por la altura de la figura en ese punto; equivale a buscar el punto donde la figura quedaría en equilibrio si fuese una placa con peso.

SÍMBOLO	SIGNIFICADO
z^*	Valor defuzzificado: la propina final.
$\mu(z)$	Grado de pertenencia de la figura agregada en cada punto z .
numerador	Suma de cada posición ponderada por su «altura» (el momento).
denominador	Área total de la figura.

Calculemos el centroide de nuestra figura. Conviene partirla en tramos geométricos sencillos y sumar el área de cada uno y su momento (área por la posición de su propio centro). La figura tiene cinco tramos:

TRAMO	FORMA	ÁREA $\int MDZ$	MOMENTO $\int Z MDZ$
[0; 7,5]	triángulo que sube a 0,6	2,250	11,250
[7,5; 17,5]	rectángulo de altura 0,6	6,000	75,000
[17,5; 20]	rectángulo con muesca hacia 0,5	1,375	25,781
[20; 25]	rectángulo de altura 0,6	3,000	67,500
Total		12,625	179,531

Dividiendo el momento total entre el área total:

$$z^* = \frac{179,531}{12,625} \approx 14,22$$

El controlador difuso recomienda una propina de aproximadamente **14,2 %**. Y fíjate en lo que ha pasado: un servicio de 7 y una comida de 8 no han disparado ni la propina mínima ni la máxima, sino un valor intermedio razonable, mezclando suavemente «media» y «alta» según cuánto se activó cada regla. Ningún umbral brusco; una transición continua. Eso es exactamente lo que la lógica binaria no sabía hacer y lo que hace tan útiles a estos controladores.

MÉTODO DE DEFUZZIFICACIÓN	QUÉ DEVUELVE	CARÁCTER
Centroide	Centro de gravedad de la figura.	Suave, tiene en cuenta toda la forma.
Bisector	El z que parte el área en dos mitades iguales.	Parecido al centroide, algo más barato.
Máximo (media)	Promedio de los puntos donde μ es máxima.	Brusco, ignora la forma de las faldas.

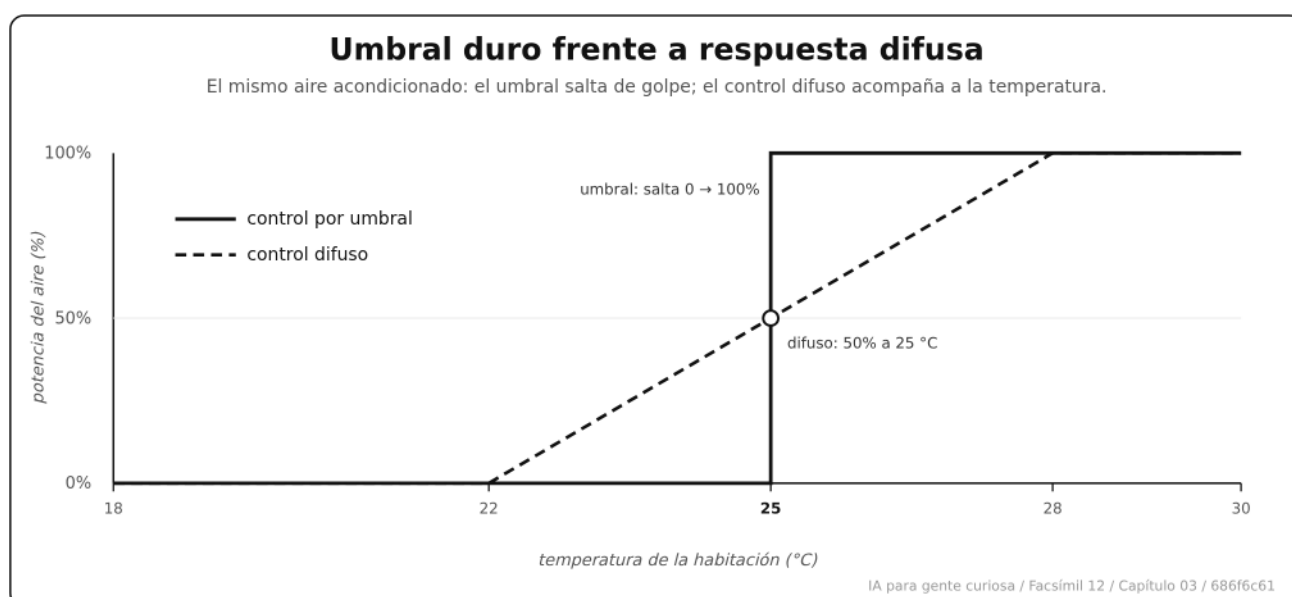
En el día a día

Aunque haya nacido en un laboratorio, la lógica difusa lleva décadas dentro de electrodomésticos que probablemente tienes en casa. Es uno de esos casos en los que una idea académica se volvió tan común que dejó de mencionarse. Donde más triunfó fue en el **control**, justo el terreno que conecta con la robótica y los sistemas embebidos.

APARATO	ENTRADAS DIFUSAS	SALIDA CONTROLADA
Aire acondicionado	temperatura, ritmo de cambio	potencia del compresor
Lavadora	suciedad del agua, tamaño de la carga	duración del ciclo, agua
Cámara de fotos	contraste de la escena, movimiento	enfoque, estabilización
Tren metropolitano	velocidad, distancia a la parada	aceleración y frenado suaves
Riego de jardín	humedad del suelo, previsión de lluvia	caudal y duración del riego
Frenos ABS	deslizamiento de la rueda, velocidad	presión de frenado

La razón por la que arrasó en control es la que vimos en el ejemplo de la propina: un controlador difuso pasa de un estado a otro sin escalones bruscos. Un termostato binario enciende y apaga de golpe, y notas el vaivén; uno difuso ajusta la potencia de forma gradual y mantiene la habitación más estable con menos consumo. El metro de Sendai, en Japón, fue un caso célebre de frenado difuso tan suave que los pasajeros apenas lo notaban.

El siguiente diagrama enfrenta las dos formas de mandar sobre el mismo aire acondicionado. El control por umbral mantiene el aparato apagado hasta los 25 °C y entonces lo dispara al 100 %: justo en la frontera, una décima de grado decide entre nada y todo, y el aparato se pasa el rato encendiéndose y apagándose. El control difuso, en cambio, empieza a soplar flojito en cuanto la habitación se entibia y va subiendo la potencia conforme sube la temperatura, sin saltos.



La misma lógica aparece en sitios menos visibles. Un sistema de **riego** automático no abre la válvula a tope en cuanto el suelo baja de un umbral: mide la humedad de la tierra y la previsión de lluvia y gradúa el caudal, de modo que un día algo seco recibe un riego suave en lugar de un encharcamiento. Los **frenos ABS** de muchos coches modulan la presión según cuánto desliza la rueda y a qué velocidad va, buscando el punto justo entre el bloqueo y la frenada timorata. Y una **cámara de fotos** ajusta enfoque y estabilización combinando el contraste de la escena y el movimiento detectado, dos entradas que nunca son del todo «sí» o «no». En todos estos casos la gracia es la misma del termostato: cambiar un interruptor brusco por una respuesta que acompaña al mundo en lugar de pelearse con él.

En el mundo del software de negocio, la lógica difusa vive hoy en las **reglas graduales**: puntuaciones de riesgo, sistemas de recomendación con preferencias suaves, motores de decisión donde «cliente muy fiel» o «pedido bastante urgente» se modelan como grados en lugar de umbrales secos. Donde no encaja especialmente es en los grandes modelos de lenguaje: un LLM aprende sus propias representaciones internas y no razona con funciones de pertenencia explícitas. Pero su primo cercano, la idea de **guardrails graduales** que vimos en el Facsímil 2, sí bebe de aquí: una política que no sea «permitido / prohibido» a secas, sino «permitido con grado de confianza», es una política difusa con otro nombre.

Por qué debería importarte

Porque la mayoría de las decisiones reales no son binarias y, sin embargo, seguimos programándolas con `if` rígidos que esconden umbrales arbitrarios. «Si la temperatura supera 25, enciende el aire» convierte una realidad gradual en un interruptor, y luego nos extraña que el sistema oscile o se comporte de forma tosca justo en la frontera.

La lógica difusa te da un vocabulario para razonar con grados sin renunciar a obtener un número concreto al final. Y los factores de certeza te enseñan que se puede asignar confianza a una conclusión y combinar evidencias sin montar toda la maquinaria probabilística. Las dos ideas comparten una lección que recorre este facsímil entero: entre el «sí» y el «no» hay un territorio enorme, y modelarlo explícitamente casi siempre produce sistemas más estables y más fáciles de explicar que fingir que ese territorio no existe.

Dónde solía tropezar yo

ERROR	POR QUÉ ES UN ERROR	ANTÍDOTO
Confundir grado de pertenencia con probabilidad	Que algo pertenezca a «caliente» con 0,7 no significa que haya un 70 % de probabilidad de que sea caliente. Son cosas distintas y los números no suman 1.	Pregúntate si la duda es «¿ocurrirá?» (probabilidad) o «¿cuánto se cumple?» (difusa).
Diseñar etiquetas que no se solapan	Si «templado» acaba justo donde empieza «caliente», recuperas los saltos bruscos que querías evitar.	Haz que las funciones de pertenencia se monten unas sobre otras en las fronteras.
Olvidar normalizar las escalas de entrada	Como en el k-means del Facsímil 1, una entrada en miles y otra entre 0 y 1 distorsionan las reglas.	Define cada universo de discurso con su rango real antes de fuzzificar.
Encadenar factores de certeza sin medir	Los CF pierden sentido cuando se encadenan muchas reglas dependientes; la creencia final puede no significar nada.	Para cadenas largas o evidencias correlacionadas, valora una red bayesiana del capítulo 2.
Elegir el método de defuzzificación al azar	El centroide y el método del máximo pueden dar valores muy distintos sobre la misma figura.	Elige el método según si quieres suavidad (centroide) o decisión tajante (máximo).

Cómo encaja todo

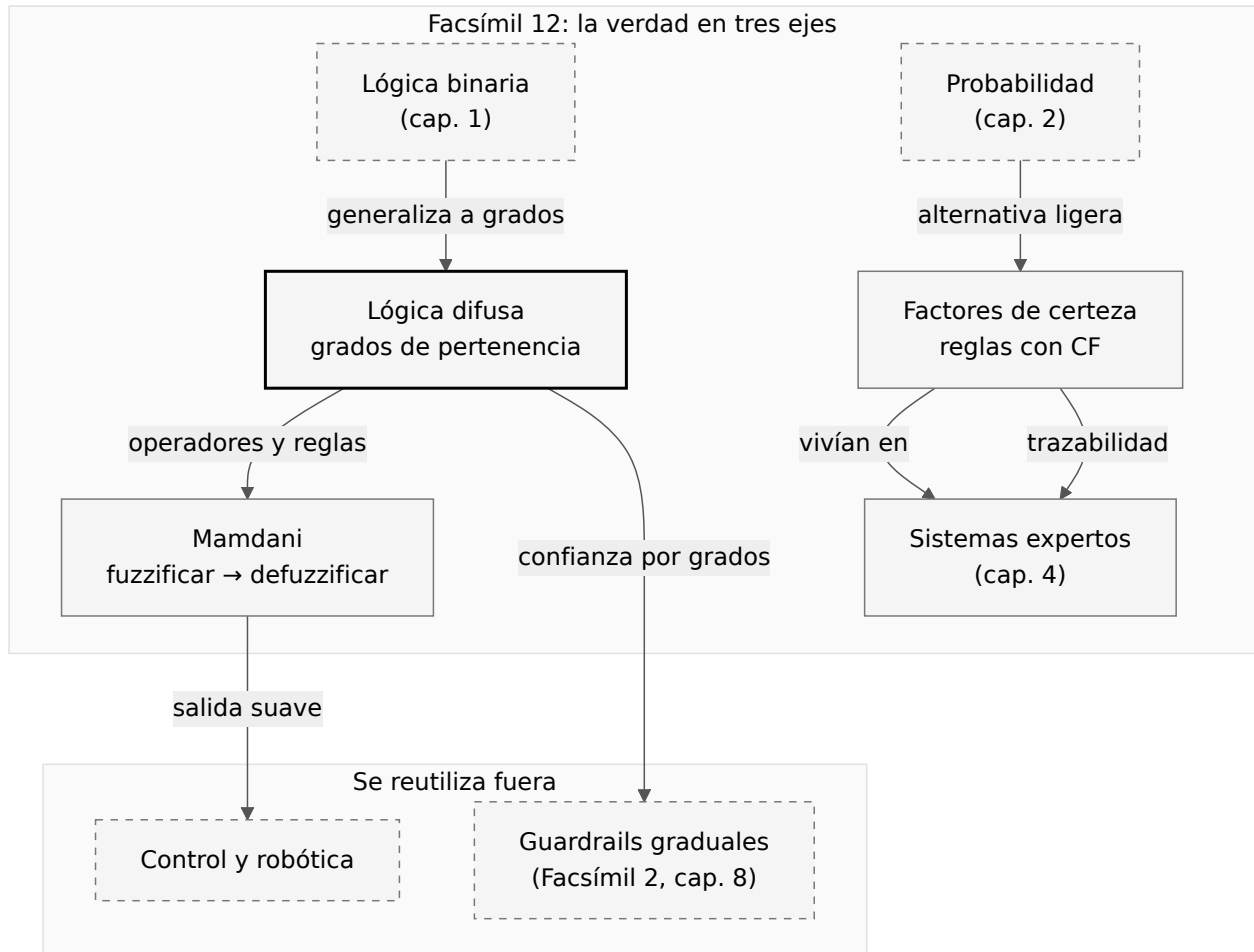
Este capítulo cierra un triángulo, no una escalera, y la diferencia importa. El capítulo 1 de este Facsímil 12 nos dio la lógica binaria, donde una afirmación es verdadera o falsa sin matices. El capítulo 2 nos dio la probabilidad, para los días en que no sabemos qué pasará. Y aquí hemos puesto el tercer vértice, la imprecisión, para cuando la propia frase admite grados. Ninguno destrona a los otros: un buen ingeniero sabe que son herramientas distintas para preguntas distintas, igual que un cocinero no tira el cuchillo por tener una batidora.

La conexión más fina, y la que más se confunde, es con ese capítulo 2. Allí, la probabilidad de que mañana llueva mide tu ignorancia sobre un hecho que será nítido: lloverá o no lloverá, y al asomarte a la ventana la duda se evapora. Aquí, en cambio, decir que a 22 °C la habitación está «templada con grado 0,8» no es ignorancia que vaya a resolverse: por mucho que mires el termómetro, 22 °C seguirá siendo templado a medias. Por eso los grados de pertenencia no suman 1 y las probabilidades sí. Cuando alguien te diga «hay un 70 % de que esto sea caliente», párate un segundo: casi siempre quiere decir «esto pertenece a caliente con grado 0,7», y son cosas distintas. Llevar la herramienta equivocada a ese problema es como medir un trozo de tela con un cronómetro.

Los **factores de certeza** miran en la otra dirección, hacia el capítulo 4 de este mismo facsímil. No nacieron como teoría elegante, sino dentro de los sistemas expertos: MYCIN, el consultor de infecciones de Stanford, necesitaba combinar reglas que un médico nunca habría firmado como verdades absolutas («si hay fiebre, *tiendo a pensar* en gripe»). Ese «tiendo a pensar» es un CF, y la cadena de reglas que MYCIN encadenaba para llegar a una conclusión es exactamente lo que el capítulo 4 celebra como explicabilidad: un sistema que no solo decide, sino que puede enseñarte el camino que recorrió. Cuando hoy una herramienta te muestra «recomendado con un 82 % de confianza, porque A y porque B», estás viendo el nieto lejano de aquellos factores de certeza.

Hacia fuera del facsímil, la lógica difusa enlaza con dos sitios que parecen lejanos pero comparten raíz. El primero son los **guardrails graduales** del capítulo 8 del Facsímil 2. Una política de seguridad que solo sepa decir «permitido» o «prohibido» se comporta como aquel termostato de un solo umbral: salta de golpe y se equivoca justo en la frontera. Una política que diga «permitido con grado de confianza» y deje pasar, bloquear o pedir revisión según ese grado es, sin más, una política difusa con otro nombre; el solapamiento entre etiquetas que dibujamos para frío y templado es el mismo que evita que un filtro de contenidos trate igual un caso clarísimo y uno dudoso. El segundo sitio, y donde la lógica difusa de verdad ganó la partida, es el **control en electrodomésticos y robótica**. La transición suave que calculamos para la propina es la misma que mantiene estable la temperatura de un aire acondicionado sin el vaivén de encender y apagar, la que ajusta el ciclo de una lavadora a lo sucia que está el agua, o la que frenaba el metro de Sendai con tanta delicadeza que los pasajeros no se agarraban a las barras. En todos esos casos no hubo una red neuronal ni un modelo gigantesco: hubo un puñado de reglas en lenguaje casi natural y un centroide, como el que hemos calculado a mano. Esa es, quizá, la mejor lección del capítulo: a veces la idea más útil no es la más sofisticada, sino la que se atreve a tomarse en serio el «más o menos».

Facsímil 12: la verdad en tres ejes



Vocabulario aprendido

TÉRMINO	DEFINICIÓN
Lógica difusa	Lógica de grados de verdad entre 0 y 1, no solo verdadero o falso.
Conjunto difuso	Conjunto donde cada elemento pertenece con un grado entre 0 y 1.
Función de pertenencia	Función que asigna a cada valor su grado de pertenencia a un conjunto difuso.
Grado de verdad	Valor entre 0 y 1 que cuantifica cuánto se cumple una afirmación difusa.
Variable lingüística	Variable cuyos valores son etiquetas como «bajo», «medio» o «alto».
Factor de certeza (CF)	Medida entre -1 y 1 del grado de creencia en una afirmación, usada en MYCIN.
Inferencia de Mamdani	Método de inferencia difusa: fuzzificar, evaluar reglas, agregar y defuzzificar.
Defuzzificación	Convertir un resultado difuso en un número concreto, por ejemplo con el centroide.
Centroide	Centro de gravedad de la figura agregada, el método de defuzzificación más usado.

Antes de pasar página

- ☐ ¿Puedo explicar la diferencia entre incertidumbre e imprecisión con un ejemplo de cada una? (Si no, vuelve a «Dos cosas distintas que llamamos incierto».)
- ☐ ¿Sé combinar dos factores de certeza positivos y entiendo por qué nunca superan 1? (Si no, vuelve a «Factores de certeza».)
- ☐ ¿Puedo dibujar una función de pertenencia triangular y leer el grado de un valor concreto? (Si no, vuelve a «Conjuntos difusos».)
- ☐ ¿Recuerdo qué operador difuso corresponde a Y, a O y a NO? (Si no, vuelve a «Operadores y reglas difusas».)
- ☐ ¿Sé recorrer los cuatro pasos de Mamdani: fuzzificar, evaluar, agregar y defuzzificar? (Si no, vuelve a «Inferencia de Mamdani».)
- ☐ ¿Entiendo qué calcula el centroide y por qué da un valor suave? (Si no, vuelve a «Paso 4: defuzzificación por centroide».)
- ☐ ¿He ejecutado el cuaderno y comprobado que el centroide ronda el 14,2 %?

En resumen

IDEA FUERZA	DETALLE
Incertidumbre e imprecisión no son lo mismo.	La probabilidad mide si un hecho nítido ocurre; la lógica difusa mide cuánto se cumple un predicado vago.
Los factores de certeza fueron el apaño práctico de los sistemas expertos.	Un CF entre -1 y 1 combina evidencias sin la maquinaria bayesiana completa, a cambio de un fundamento más débil.
Un conjunto difuso pertenece a medias.	La función de pertenencia da grados entre 0 y 1 , y las etiquetas solapadas convierten números en palabras.
Mamdani transforma reglas en números.	Fuzzificar, evaluar reglas con min y max, agregar y defuzzificar producen una salida concreta y suave.
La lógica difusa brilla en control.	Termostatos, lavadoras y trenes la usan porque evita los saltos bruscos de la lógica binaria.

Para saber más

Buchanan, B. G. y Shortliffe, E. H. (1984). *Rule-based expert systems: The MYCIN experiments of the Stanford Heuristic Programming Project*. Addison-Wesley.

Mamdani, E. H. y Assilian, S. (1975). An experiment in linguistic synthesis with a fuzzy logic controller. *International Journal of Man-Machine Studies*, 7(1), 1-13.

[https://doi.org/10.1016/S0020-7373\(75\)80002-2](https://doi.org/10.1016/S0020-7373(75)80002-2)

Ross, T. J. (2010). *Fuzzy logic with engineering applications* (3.^a ed.). Wiley.

Russell, S. y Norvig, P. (2021). *Artificial intelligence: a modern approach* (4.^a ed.). Pearson.

Shortliffe, E. H. (1976). *Computer-based medical consultations: MYCIN*. Elsevier.

Zadeh, L. A. (1965). Fuzzy sets. *Information and Control*, 8(3), 338-353.

[https://doi.org/10.1016/S0019-9958\(65\)90241-X](https://doi.org/10.1016/S0019-9958(65)90241-X)

Notas

1. Zadeh, L. A. (1965). Fuzzy sets. *Information and Control*, 8(3), 338-353.

[https://doi.org/10.1016/S0019-9958\(65\)90241-X](https://doi.org/10.1016/S0019-9958(65)90241-X) ↵

2. Russell, S. y Norvig, P. (2021). *Artificial intelligence: a modern approach* (4.ª ed.). Pearson. El capítulo sobre razonamiento con incertidumbre separa explícitamente la vaguedad (lógica difusa) de la incertidumbre probabilística, y advierte de los errores que surgen al confundirlas. ↩
 3. Shortliffe, E. H. (1976). *Computer-based medical consultations: MYCIN*. Elsevier. La obra describe el sistema MYCIN y, en particular, el modelo de factores de certeza diseñado para razonar con reglas médicas que no eran lógicamente concluyentes. ↩
 4. Buchanan, B. G. y Shortliffe, E. H. (1984). *Rule-based expert systems: The MYCIN experiments of the Stanford Heuristic Programming Project*. Addison-Wesley. Recopila el modelo de factores de certeza, sus reglas de combinación y la discusión sobre sus límites teóricos frente a la probabilidad. ↩
 5. Ross, T. J. (2010). *Fuzzy logic with engineering applications* (3.ª ed.). Wiley. Desarrolla las operaciones sobre conjuntos difusos, los sistemas de reglas y los métodos de inferencia y defuzzificación que se usan en control. ↩
 6. Mamdani, E. H. y Assilian, S. (1975). An experiment in linguistic synthesis with a fuzzy logic controller. *International Journal of Man-Machine Studies*, 7(1), 1-13.
[https://doi.org/10.1016/S0020-7373\(75\)80002-2](https://doi.org/10.1016/S0020-7373(75)80002-2) ↩
-

Cuadernos para practicar

Escanea el código con el móvil para abrir el cuaderno en Google Colab y ejecutarlo sin instalar nada.



Lógica difusa: cuando «verdadero» y «falso» no bastan

<https://colab.research.google.com/github/686f6c61/iaparagentecuriosa-notebooks/blob/main/Facs%C3%ADmil%2012/03-logica-difusa.ipynb>

Capítulo 04: Sistemas expertos y soporte a la decisión

Entrando en el tema

En el capítulo 1 de este facsímil vimos cómo una máquina puede representar afirmaciones en lógica proposicional y de primer orden, y derivar consecuencias por resolución. Eso es la maquinaria. Este capítulo trata de qué se construye con ella cuando el objetivo no es demostrar un teorema, sino imitar a un experto: un médico que diagnostica, un analista que aprueba o rechaza un préstamo, un técnico que localiza una avería.

Ya conociste la silueta de un sistema experto en el Facsímil 2, capítulo 12: una ontología que dice qué tipos de cosas existen, un conjunto de hechos, un conjunto de reglas, un motor de inferencia que las aplica y un módulo de explicación que justifica el resultado. Aquel capítulo se centró en el conocimiento: cómo se modela, cómo se representa con RDF y OWL, cómo se consulta con SPARQL. Aquí entramos en la pieza que allí quedó descrita en una frase y que ahora abrimos en canal: el **motor de inferencia**. Cómo decide. Qué algoritmo ejecuta. Por qué a veces parte de los datos y a veces parte de la conclusión que quiere confirmar.

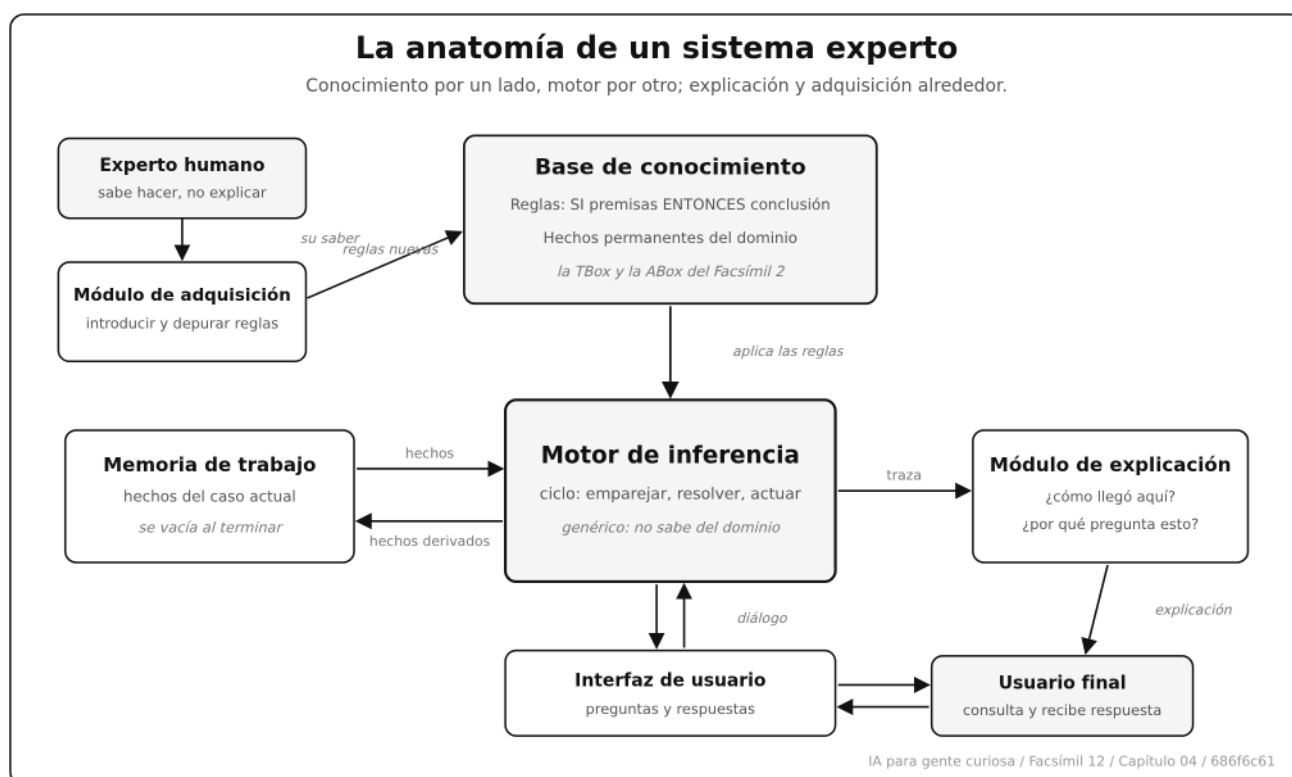
No es arqueología. Los sistemas expertos vivieron su edad de oro entre los años setenta y ochenta, pero su mecánica reaparece hoy, intacta, cada vez que un agente con un modelo de lenguaje consulta una tabla de permisos antes de ejecutar una acción, o cada vez que un *guardrail* aplica una regla de negocio que no puede dejarse al criterio probabilístico de una red neuronal. Una regla que dice «si el importe supera el límite, escala a una persona» no necesita un modelo de mil millones de parámetros. Necesita dispararse siempre, igual, y poder explicarse. Ese «siempre, igual, y explicable» es el corazón de este capítulo.

La arquitectura de un sistema experto

Un sistema experto separa deliberadamente dos cosas que en un programa normal van mezcladas: **qué sabe** y **cómo razona**. El conocimiento del dominio vive en la base de conocimiento, declarado como hechos y reglas; el mecanismo de razonamiento vive en el motor de inferencia, que es genérico y no sabe nada de medicina ni de banca.¹ Esa separación es la idea de ingeniería más importante de todo el paradigma: puedes cambiar las reglas sin tocar el motor, y reutilizar el motor en otro dominio sin reescribirlo.

COMPONENTE	QUÉ CONTIENE	QUIÉN LO ESCRIBE
Base de conocimiento	Hechos permanentes y reglas del dominio.	El ingeniero del conocimiento, con el experto.
Memoria de trabajo	Hechos conocidos del caso actual.	El motor, durante una consulta.
Motor de inferencia	El algoritmo que aplica reglas a hechos.	El fabricante del sistema, una vez.
Módulo de explicación	El rastro de qué reglas se dispararon y por qué.	El motor, registrando su propia traza.
Módulo de adquisición	Las herramientas para introducir y depurar reglas.	El equipo, de forma continua.
Interfaz de usuario	El diálogo de preguntas y respuestas.	El equipo, según el usuario final.

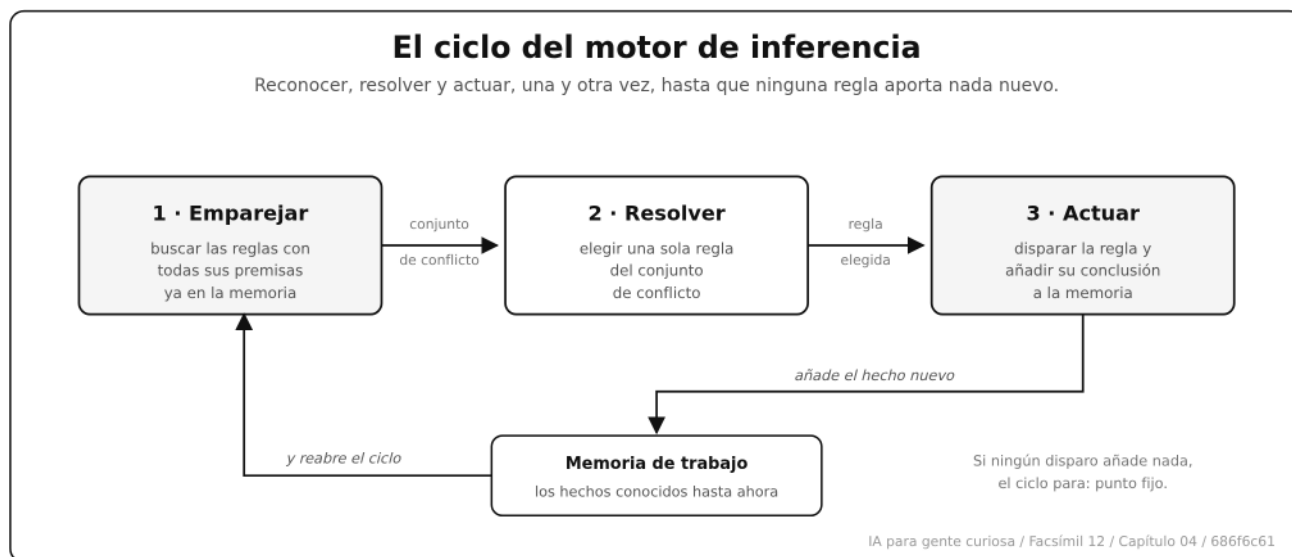
Antes de entrar en cada pieza, conviene verlas juntas. El siguiente diagrama coloca el motor de inferencia en el centro y dispone a su alrededor el resto de módulos: la base de conocimiento que lo alimenta, la memoria de trabajo con la que intercambia hechos, la explicación que produce y la adquisición por la que entra el saber del experto.



La **base de conocimiento** es la unión de dos cosas que en el Facsímil 2 llamamos TBox y ABox: el modelo del dominio y los hechos permanentes. Aquí añadimos las **reglas de producción**, escritas como **SI condición ENTONCES conclusión**. Una regla no es un dato: es una porción de razonamiento congelado, la frase que un experto repetiría una y otra vez sin pensar.

La **memoria de trabajo** es el cuaderno de notas del caso concreto. Empieza con lo que el usuario aporta —«ingresos de 2.400 euros», «sin impagos»— y crece a medida que el motor deriva hechos nuevos. Cuando termina la consulta, se borra: el siguiente cliente empieza con el cuaderno en blanco, pero con la misma base de conocimiento.

El **motor de inferencia** es el protagonista de este capítulo. Su trabajo es un ciclo de tres pasos que se repite hasta que no hay nada más que hacer: emparejar reglas con hechos, elegir cuál disparar y ejecutarla.



El **módulo de explicación** es lo que distingue a un sistema experto de una caja negra. Como el motor registra cada regla que dispara, puede contestar a dos preguntas que un usuario siempre acaba haciendo: «¿cómo has llegado a esto?» (mostrando la cadena de reglas) y «¿por qué me preguntas esto?» (mostrando la regla cuya premisa está intentando comprobar). Esa trazabilidad no es un adorno. Es la razón por la que, todavía hoy, una decisión de crédito o un diagnóstico clínico apoyado en reglas se puede auditar, y una decisión de una red neuronal opaca, no tanto.

El cuello de botella del conocimiento

Queda un componente que no aparece en el diagrama porque no es código: la **adquisición del conocimiento**. Alguien tiene que sentarse con el experto y traducir su saber a reglas. Y ahí está el problema histórico de todo el paradigma.

El experto sabe hacer su trabajo, pero rara vez sabe explicar las reglas que sigue. Un médico veterano reconoce una infección «de un vistazo»; pedirle que enuncie las cien condiciones que su intuición evalúa en un segundo es lento, incompleto y frustrante. A este atasco se le llama el **cuello de botella de la adquisición del conocimiento**, y fue, más que ninguna limitación técnica, lo que frenó a los sistemas expertos.² Construir el motor era barato; llenarlo de reglas correctas, caro y eterno.

El motor de inferencia: dos formas de razonar

Hay dos maneras de aplicar las mismas reglas, y la diferencia entre ellas es la espina dorsal de este capítulo. Una parte de lo que sabes y avanza hacia lo que puedes concluir. La otra parte de lo que quieres concluir y retrocede hacia lo que necesitarías saber. Se llaman **encadenamiento hacia delante** y **encadenamiento hacia atrás**, y vamos a verlas sobre la misma base de reglas para que el contraste sea nítido.

La base de reglas que usaremos

Tomemos un caso reconocible: la preevaluación de una solicitud de préstamo. No es el sistema real de un banco —esos tienen cientos de reglas y modelos estadísticos por debajo—, pero captura la forma exacta del razonamiento. Las proposiciones son afirmaciones que un analista marcaría como ciertas o no.

REGLA	SI SE CUMPLE...	...ENTONCES SE CONCLUYE
R1	<code>nomina_domiciliada</code> $\wedge$ <code>antiguedad_2anios</code>	<code>empleo_estable</code>
R2	<code>empleo_estable</code> $\wedge$ <code>sin_impagos</code>	<code>solvencia_demostrada</code>
R3	<code>solvencia_demostrada</code> $\wedge$ <code>ratio_deuda_bajo</code>	<code>riesgo_bajo</code>
R4	<code>riesgo_bajo</code> $\wedge$ <code>importe_dentro_limite</code>	<code>aprobar</code>
R5	<code>ratio_deuda_alto</code>	<code>riesgo_alto</code>
R6	<code>riesgo_alto</code>	<code>escalar_a_analista</code>

Y los hechos que el solicitante aporta al empezar, la memoria de trabajo inicial:

$$MT_0 = \{ \text{nomina_domiciliada}, \text{antiguedad_2anios}, \text{sin_impagos}, \text{ratio_deuda_bajo}, \text{importe_dentro} \}$$

En palabras: la memoria de trabajo de partida es el conjunto de los cinco hechos que el solicitante aporta antes de que el motor razone nada.

La pregunta del caso es siempre la misma: **¿se aprueba el préstamo?**, es decir, ¿podemos derivar `aprobar` ? Vamos a contestarla de las dos maneras.

Encadenamiento hacia delante: de los datos a la conclusión

El encadenamiento hacia delante está **dirigido por los datos**. Mira lo que hay en la memoria de trabajo, busca toda regla cuyas premisas estén satisfechas, la dispara, añade su conclusión a la memoria y repite. No tiene una meta en mente: deriva todo lo que se puede derivar y, de paso, llega a la conclusión que nos interesa. Es la forma natural cuando llegan datos y quieres ver «qué se sigue de esto».³

```
ALGORITMO encadenamiento_hacia_delante(reglas, MT):
  repetir:
    conjunto_conflicto <- []
    para cada regla R en reglas:
      si premisas(R) estan todas en MT y conclusion(R) NO esta en MT:
        añadir R a conjunto_conflicto
    si conjunto_conflicto esta vacio:
      terminar                                     # punto fijo: nada nuevo que derivar
    R* <- resolver_conflictos(conjunto_conflicto)
    MT <- MT u { conclusion(R*) }                 # disparar la regla elegida
    registrar(R*)                                # para la explicación
  devolver MT
```

Sigamos la traza sobre nuestro caso. En cada ciclo se muestra qué reglas están listas, cuál se dispara y qué hecho nuevo entra en la memoria.

CICLO	CONJUNTO DE CONFLICTO	REGLA DISPARADA	HECHO AÑADIDO A LA MEMORIA DE TRABAJO
1	R1	R1	<code>empleo_estable</code>
2	R2	R2	<code>solvencia_demostrada</code>
3	R3	R3	<code>riesgo_bajo</code>
4	R4	R4	<code>aprobar</code>
5	(vacío)	—	nada nuevo: alto

Fíjate en los detalles. R5 nunca entra en el conjunto de conflicto porque `ratio_deuda_alto` no está en la memoria y nunca se deriva; sin esa premisa, R6 tampoco se activa. El motor no «descarta» el rechazo: sencillamente, nada lo sostiene. Al final del ciclo 4 ya está `aprobar` en la memoria, pero un motor hacia delante puro sigue intentándolo en el ciclo 5 hasta confirmar que no queda nada por derivar: **ha alcanzado un punto fijo**. La memoria final es:

$$MT_{\text{final}} = MT_0 \cup \{\text{empleo_estable, solvencia_demostrada, riesgo_bajo, aprobar}\}$$

En palabras: la memoria final reúne los cinco hechos de partida más los cuatro que el motor derivó por el camino, incluido el `aprobar` que buscábamos.

Derivó cuatro hechos nuevos, uno de los cuales era el que buscábamos. Esa es la firma del encadenamiento hacia delante: produce **todo** lo que se sigue de los datos, lo necesites o no.

Encadenamiento hacia atrás: de la hipótesis a los hechos

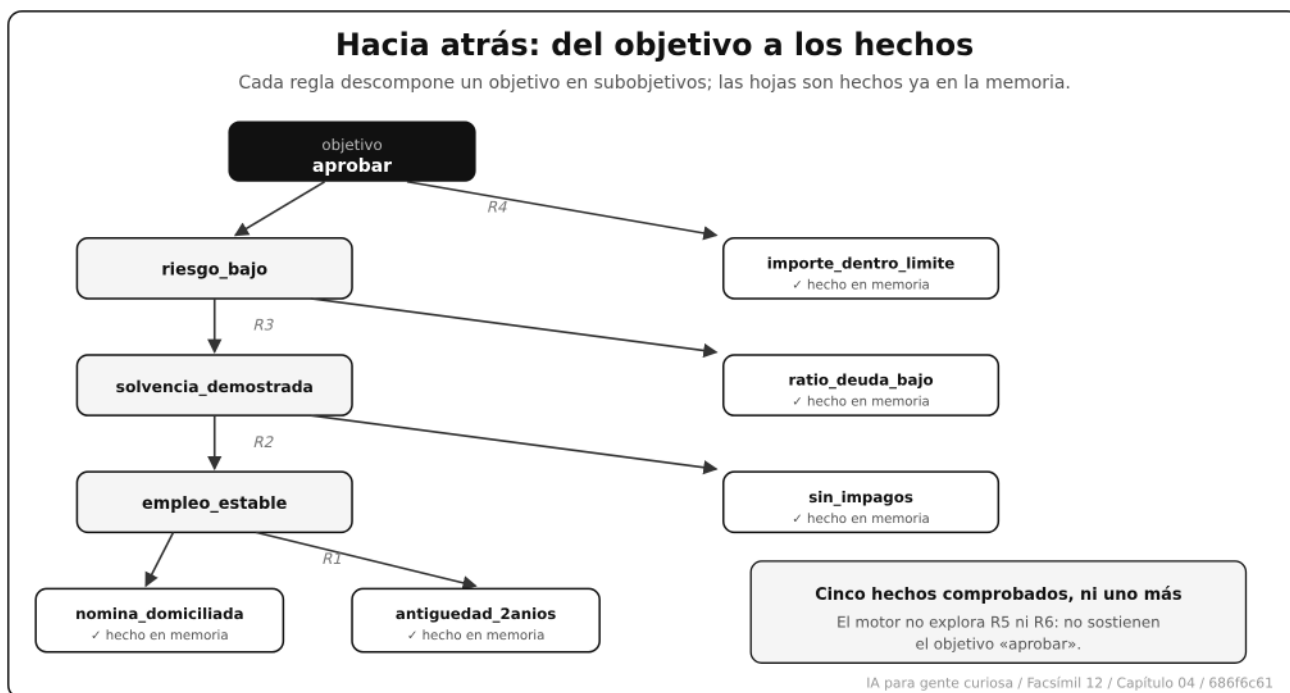
El encadenamiento hacia atrás está **dirigido por objetivos**. Parte de la conclusión que quiere confirmar —aquí, `aprobar`— y se pregunta: ¿qué regla la produce? La de R4. ¿Y qué necesitaría R4? Que se cumplan sus premisas. Convierte cada premisa en un nuevo objetivo y repite el proceso hacia atrás, hasta tocar hechos que ya están en la memoria de trabajo o que puede preguntar al usuario. Es la forma natural cuando tienes una hipótesis y quieres saber si se sostiene: el modo en que razona un médico que sospecha una enfermedad concreta y va pidiendo solo las pruebas que la confirmarían.⁴

```
ALGORITMO probar(objetivo, reglas, MT):
  si objetivo esta en MT:
    devolver VERDADERO                # ya es un hecho conocido
  para cada regla R cuya conclusion sea objetivo:
    si para toda premisa P de R: probar(P, reglas, MT) es VERDADERO:
      registrar(R)                    # esta regla sostiene el objetivo
      devolver VERDADERO
  devolver FALSO                      # ninguna via lo sostiene
```

La traza es un árbol de subobjetivos. Lo recorremos en profundidad; la sangría indica qué objetivo depende de cuál.

PROFUNDIDAD	OBJETIVO	CÓMO SE RESUELVE
0	aprobar	lo produce R4 → probar sus premisas
1	· riesgo_bajo	lo produce R3 → probar sus premisas
2	·· solvencia_demostrada	lo produce R2 → probar sus premisas
3	··· empleo_estable	lo produce R1 → probar sus premisas
4	···· nomina_domiciliada	está en la memoria ✓
4	···· antiguedad_2anios	está en la memoria ✓
3	··· sin_impagos	está en la memoria ✓
2	·· ratio_deuda_bajo	está en la memoria ✓
1	· importe_dentro_limite	está en la memoria ✓
0	aprobar	todas las ramas verdaderas → probado

El resultado es el mismo — aprobar —, pero el camino es radicalmente distinto. El motor hacia atrás nunca tocó R5 ni R6: no le hizo falta, porque no contribuyen a su objetivo. Tampoco derivó hechos «de paso». Exploró solo la parte del grafo de reglas que sostiene la hipótesis. Si en lugar de aprobar hubiéramos preguntado por `escalar_a_analista`, habría bajado por R6 y R5, habría encontrado que `ratio_deuda_alto` no está en la memoria, y habría devuelto falso sin tocar R1-R4.



Cuándo usar cada uno

No hay un ganador: hay un encaje según la forma del problema. La tabla resume el contraste que acabamos de trazar.

CRITERIO	HACIA DELANTE (DATOS)	HACIA ATRÁS (OBJETIVOS)
Punto de partida	Los hechos conocidos.	La hipótesis a confirmar.
Qué deriva	Todo lo derivable.	Solo lo que sostiene la meta.
Reglas tocadas en el caso	R1-R4 (R5, R6 nunca disparan).	R4, R3, R2, R1 (ignora R5, R6).
Bueno cuando	Llegan datos y quieres ver qué implican.	Tienes una sospecha concreta.
Riesgo	Derivar mucho que no usarás.	No te enteras de conclusiones colaterales.
Ejemplo típico	Monitorización, alarmas, configuración.	Diagnóstico, consulta puntual.

El diagrama lo resume de un vistazo: las mismas reglas, recorridas en sentidos opuestos. A la izquierda, los hechos empujan hacia arriba hasta **aprobar**; a la derecha, el objetivo **aprobar** tira hacia abajo hasta los hechos.

Las mismas reglas, recorridas al revés

A la izquierda manda el dato; a la derecha, la meta. Ambos llegan a «aprobar».

Hacia delante
datos → conclusión



Deriva 4 hechos nuevos. R5 y R6 nunca disparan.

Hacia atrás
objetivo → hechos



Comprueba 5 hechos. Ignora R5 y R6.
IA para gente curiosa / Facsímil 12 / Capítulo 04 / 686f6c61

Muchos sistemas reales combinan ambos: usan encadenamiento hacia delante para mantener actualizado un panel de hechos derivados y hacia atrás para contestar preguntas concretas del usuario sin recalcularlo todo.

Factores de certeza: arrastrar la incertidumbre por la cadena

Hasta aquí cada hecho era blanco o negro: `sin_impagos` estaba en la memoria o no estaba, y la conclusión `aprobar` salía como un sí rotundo. Pero un experto rara vez trabaja con certezas absolutas. El analista cree que la nómina está domiciliada «con bastante seguridad», porque lo ha visto en un extracto que podría estar desactualizado; el médico de MYCIN sospechaba una infección «con cierta confianza», no con una prueba irrefutable. Para que un sistema experto arrastre esa duda en lugar de fingir que no existe, MYCIN introdujo los **factores de certeza**: un número entre -1 y 1 que acompaña a cada hecho y a cada regla.⁵ Un factor de certeza de 1 es certeza total a favor; de -1, certeza en contra; de 0, ignorancia.

Conviene ser honestos sobre lo que es y lo que no es. El esquema de factores de certeza no es una probabilidad rigurosa: sus propios autores lo presentaron como una heurística práctica para cuando no hay probabilidades fiables a mano, y los capítulos 2 y 3 de este facsímil formalizan la incertidumbre con la teoría de la probabilidad que este modelo solo aproximaba.⁶ Aun así captura una intuición correcta: una conclusión derivada de premisas dudosas, encadenadas una tras otra, no puede ser más firme que el más débil de sus eslabones, y se va desgastando a cada paso. Tres reglas de combinación bastan para propagarlo por toda la base.

La primera gobierna la **conjunción de premisas**. Cuando una regla exige varias condiciones unidas por «Y», su premisa es tan firme como la condición más floja.

$$CF(P_1 \wedge P_2 \wedge \dots \wedge P_n) = \min_i CF(P_i)$$

En palabras: la certeza de una premisa con varias condiciones unidas por «Y» es la menor de las certezas de esas condiciones; la cadena vale lo que vale su eslabón más débil.

La segunda gobierna la **propagación por una regla**. Una regla traslada a su conclusión la certeza de su premisa, rebajada por la confianza que merece la propia regla.

$$CF(H) = CF(\text{premisa}) \times CF_{\text{regla}}$$

En palabras: la certeza del hecho concluido es la certeza de la premisa multiplicada por el factor de certeza de la regla; como ambos valen como mucho 1, la conclusión sale siempre igual o menos firme que la premisa de la que nace.

La tercera gobierna la **combinación de dos reglas que concluyen lo mismo**. Si dos reglas distintas apoyan el mismo hecho con evidencia independiente y a favor, sus certezas se suman, pero descontando el solapamiento para no pasarse de 1.

$$CF(H) = CF_1 + CF_2 - CF_1 \cdot CF_2$$

En palabras: dos pruebas independientes que apuntan al mismo hecho refuerzan la creencia, pero el resultado nunca alcanza el 1; cada prueba nueva cubre parte del terreno que la anterior dejaba en duda, no todo.

SÍMBOL O	SIGNIFICADO	EJEMPLO
CF	Factor de certeza: grado de creencia entre -1 y 1.	CF = 1, certeza total a favor; CF = 0, ignorancia; CF = -1, certeza en contra.
CF(premisa)	Certeza de la condición completa de una regla.	el mínimo de las certezas de cada condición unida por «Y».
CF de la regla	Confianza que el experto deposita en la propia regla.	R1 vale 0,88: «casi siempre se cumple, pero no siempre».
CF(H)	Certeza del hecho H que la regla concluye.	CF(empleo_estable) = 0,792.
min	Operador que toma el menor de varios valores.	min(0,90; 0,85) = 0,85.
CF ₁ , CF ₂	Aportes de dos reglas independientes al mismo hecho.	dos vías distintas que apuntan a riesgo_alto .

Llevémoslo a números sobre el caso del préstamo. Los hechos del solicitante no entran al cien por cien: la nómina, la antigüedad y la ausencia de impagos valen 0,90; el ratio de deuda bajo, 0,85; el importe dentro del límite, 1,00. Y cada regla lleva su propia confianza: R1, R2 y R3 valen 0,88, y R4 vale 0,86. Aplicando las dos primeras fórmulas paso a paso por la cadena R1 → R2 → R3 → R4:

PASO	HECHO DERIVADO	CF DE LA PREMISA	× CF DE LA REGLA	CF DEL HECHO
R1	empleo_estable	$\min(0,90; 0,90) = 0,90$	× 0,88	0,792
R2	solvencia_demostrada	$\min(0,792; 0,90) = 0,792$	× 0,88	0,697
R3	riesgo_bajo	$\min(0,697; 0,85) = 0,697$	× 0,88	0,613
R4	aprobar	$\min(0,613; 1,00) = 0,613$	× 0,86	≈ 0,5275

La columna intermedia se muestra redondeada para que se lea, pero el motor encadena los valores sin redondear los pasos ($0,90 \times 0,88 \times 0,88 \times 0,86$) y solo redondea al final: por eso el factor de certeza de `aprobar` queda en **0,5275** y no un sí tajante. Fíjate en lo que ha pasado: partiendo de hechos bastante firmes (0,90) y reglas fiables (0,88), cuatro saltos encadenados han ido limando la certeza hasta dejarla en torno a la mitad. Esa es la lección que MYCIN aprendió y que cualquier sistema de reglas hereda: encadenar mucho desgasta la confianza, y una conclusión que cuelga de una cadena larga merece más cautela que una que cuelga de un solo paso. Es exactamente el número que obtienes en el cuaderno del facsímil al propagar la certeza por esta misma cadena.

La tercera fórmula entra en juego cuando dos reglas independientes empujan hacia el mismo hecho. Imagina dos indicios de `riesgo_alto`: una regla lo deriva con CF 0,6 y otra, por una vía distinta, con CF 0,7. Combinados, la creencia sube a $0,6 + 0,7 - 0,6 \times 0,7 = \mathbf{0,88}$, más que cualquiera de los dos por separado, pero sin llegar al 1, porque dos pruebas que se solapan no suman certezas a ciegas. Esta es la pieza que conecta este capítulo, hecho de reglas nítidas, con los dos siguientes, hechos de grados de creencia: la regla dice qué se concluye y el factor de certeza, con cuánta confianza.

Resolución de conflictos: elegir qué regla disparar

En nuestra traza, cada ciclo tenía una sola regla lista. En la vida real, varias reglas compiten por dispararse en el mismo ciclo: ese conjunto se llama **conjunto de conflicto**, y el motor necesita un criterio para elegir. La elección importa, porque disparar una regla cambia la

memoria y puede activar o desactivar otras. Las estrategias clásicas son cuatro, y los motores serios las combinan en cascada.⁷

ESTRATEGIA	QUÉ PREFIERE	PARA QUÉ SIRVE
A		
Refracción	No volver a disparar la misma regla con los mismos hechos.	Evita bucles infinitos.
Novedad (recency)	La regla que usa los hechos más recientes.	Sigue la línea de razonamiento en curso.
Especificidad	La regla con más premisas o más concretas.	Una excepción afinada gana a una regla general.
Prioridad (salience)	El peso que el ingeniero asigna a mano.	Garantiza que las reglas críticas van primero.

La especificidad merece un ejemplo, porque es la que captura el sentido común de «la excepción manda». Imagina dos reglas: «si es un ave, entonces vuela» y «si es un ave y es un pingüino, entonces no vuela». Ante un pingüino las dos están listas, pero la segunda es más específica y debe ganar. Sin un criterio de especificidad, el orden de disparo sería arbitrario y el sistema podría concluir que el pingüino vuela. La prioridad explícita es la red de seguridad: cuando ninguna heurística decide bien, el ingeniero fija a mano que «escalar a una persona» pese más que «aprobar automáticamente». Ese mismo patrón reaparece, con otro nombre, en los *guardrails* de los agentes modernos, donde una regla de seguridad debe imponerse siempre sobre la sugerencia del modelo.

Rete: por qué el emparejamiento no mata el rendimiento

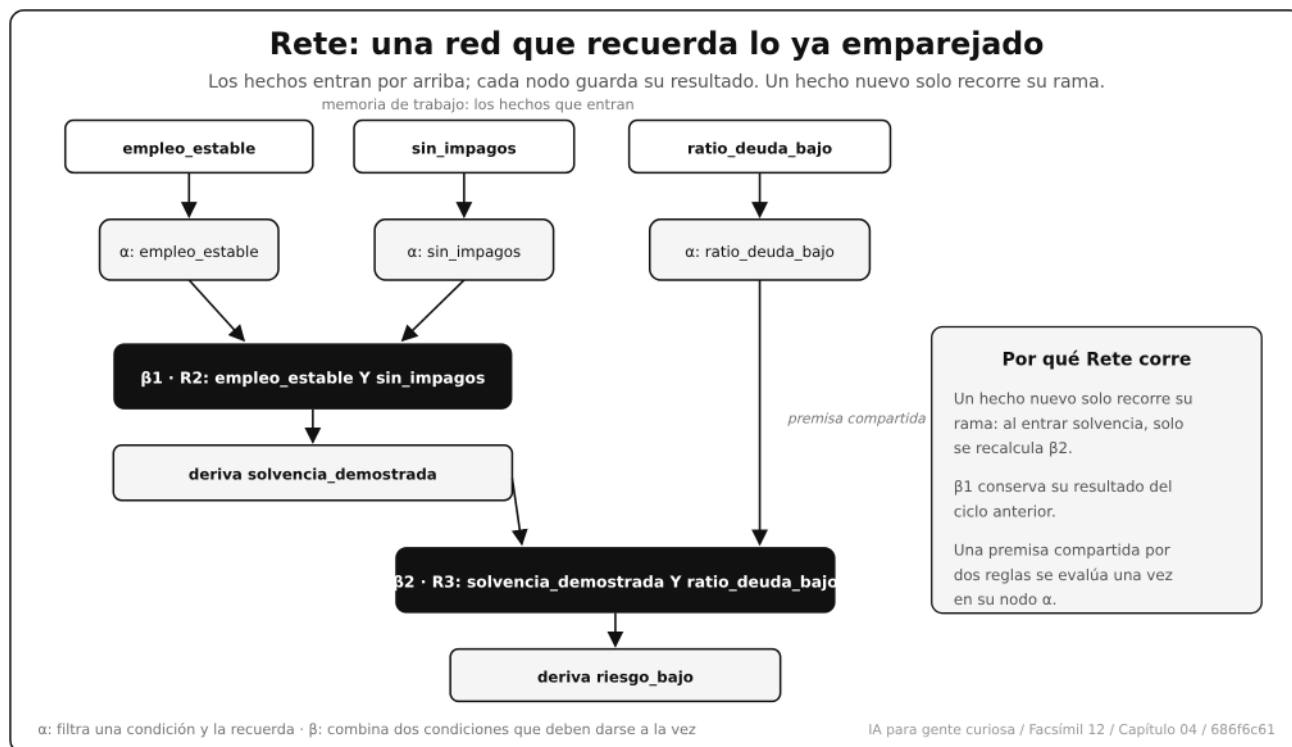
Hay un problema de eficiencia escondido en el algoritmo hacia delante. En cada ciclo recorriamos **todas** las reglas comprobando si sus premisas estaban en la memoria. Con seis reglas y cinco hechos no importa. Con diez mil reglas y cien mil hechos, reevaluarlo todo en cada ciclo es inviable: el motor pasaría más tiempo emparejando que razonando.

La observación clave que rompe el atasco es que **la memoria de trabajo cambia poco entre ciclos**: disparar una regla añade uno o dos hechos, no rehace la memoria entera. Entonces, ¿por qué recalcular desde cero qué reglas encajan? El algoritmo **Rete**, que Charles Forgy publicó en 1982, explota exactamente esa idea.⁸

Rete compila las reglas en una red de nodos por la que «fluyen» los hechos. Cada nodo comprueba una condición elemental y **recuerda** qué hechos la han pasado. Cuando entra un hecho nuevo, solo se propaga por las ramas afectadas; el resto de la red conserva sus resultados parciales del ciclo anterior. La red tiene dos partes: los nodos *alfa*, que filtran

hechos individuales por sus condiciones simples, y los nodos *beta*, que combinan hechos que deben cumplirse a la vez. Además, si varias reglas comparten una premisa, esa premisa se evalúa una sola vez y el resultado se reparte.

El diagrama traza dos reglas encadenadas de nuestra base, R2 y R3, compiladas en una red. Los hechos entran por arriba, bajan a los nodos alfa que los filtran, y se combinan en los nodos beta que activan cada regla. Lo importante es lo que ocurre cuando R2 deriva un hecho nuevo: solo recorre la rama que cuelga de él, mientras el resto de la red conserva lo que ya había emparejado.



SIN RETE

CON RETE

Cada ciclo reevalúa todas las reglas.

Solo se propaga el cambio incremental.

Las premisas compartidas se recalculan.

Las premisas compartidas se evalúan una vez.

El coste crece con el número de reglas.

El coste crece con el número de cambios.

Sencillo de programar.

Cambia memoria por velocidad.

El precio que paga Rete es memoria: guardar todos esos estados parciales ocupa espacio. Es el clásico canje de espacio por tiempo. Pero ese canje fue lo que hizo posible que sistemas como XCON, que veremos enseguida, ejecutaran miles de reglas en tiempo razonable. Rete sigue latiendo dentro de motores de reglas actuales como Drools o CLIPS.

Los sistemas expertos clásicos: qué lograron y por qué fueron frágiles

La teoría anterior no nació en el vacío. Se forjó construyendo sistemas concretos que, por primera vez, hicieron que un programa rindiera como un especialista humano en una tarea estrecha. Tres de ellos definieron la época.

DENDRAL (Universidad de Stanford, desde mediados de los sesenta) infería la estructura molecular de un compuesto químico a partir de datos de espectrometría de masas. Fue el primer gran sistema experto y demostró algo entonces sorprendente: codificar el conocimiento heurístico de químicos expertos permitía a un programa proponer estructuras plausibles que un humano tardaría mucho más en enumerar. Estableció el patrón de toda la disciplina: el rendimiento venía del **conocimiento específico del dominio**, no de un razonador general más listo.

MYCIN (Stanford, primera mitad de los setenta) diagnosticaba infecciones bacterianas de la sangre y recomendaba antibióticos. Trabajaba hacia atrás desde la hipótesis «el paciente tiene tal infección» pidiendo solo las pruebas relevantes, y manejaba la incertidumbre con **factores de certeza**, un esquema numérico para combinar evidencia parcial.⁹ En evaluaciones controladas, sus recomendaciones igualaron y a veces superaron a las de especialistas humanos. Y, sin embargo, **MYCIN nunca se usó en la práctica clínica**: las razones no fueron de calidad, sino de responsabilidad legal, de integración en el flujo hospitalario y de la incomodidad de depender de un programa para una decisión médica. El proyecto, documentado al detalle por Buchanan y Shortliffe, se convirtió en el estudio de caso fundacional sobre cómo construir y, sobre todo, cómo mantener una base de reglas.¹⁰

XCON (también llamado R1, en Digital Equipment Corporation, finales de los setenta y los ochenta) configuraba pedidos de miniordenadores VAX: comprobaba que los componentes que un cliente había pedido fueran compatibles y completos, y generaba el diagrama de montaje. Fue el primer gran éxito **comercial** de la tecnología, con un ahorro estimado de decenas de millones de dólares anuales para la empresa. Usaba encadenamiento hacia delante sobre un motor con Rete, y llegó a contener varios miles de reglas.

SISTEMA	DOMINIO	LO QUE DEMOSTRÓ	SU TALÓN DE AQUILES
DENDRAL	Estructura molecular.	El conocimiento del dominio bate al razonador general.	Dominio estrechísimo.
MYCIN	Diagnóstico de infecciones.	Una máquina podía igualar a un especialista.	Nunca se desplegó; difícil de mantener.
XCON / R1	Configuración de hardware.	Rentabilidad real a escala industrial.	Miles de reglas frágiles de mantener.

¿Por qué, si funcionaban, casi todos acabaron arrinconados? Por tres fragilidades que comparten y que conviene tener grabadas, porque son las mismas que acechan a cualquier sistema basado en reglas, también hoy:

- **El mantenimiento se vuelve infernal.** XCON pasó de unos cientos a varios miles de reglas, y cada regla nueva podía interactuar de forma imprevista con las existentes. Con miles de reglas, nadie entiende ya el sistema completo, y un cambio inocente provoca efectos colaterales lejanos. La base de conocimiento se convierte en un organismo que solo unos pocos se atreven a tocar.
- **La adquisición no termina nunca.** El cuello de botella del conocimiento no era un coste inicial, sino permanente: el mundo cambia, salen antibióticos nuevos, aparecen modelos de hardware, y alguien tiene que extraer del experto las reglas correspondientes una y otra vez.
- **No hay sentido común.** Un sistema experto solo sabe lo que está escrito en sus reglas. Fuera de su dominio estrecho es ciego: no sabe que un paciente no puede pesar 4.000 kilos ni que un pedido sin fuente de alimentación es absurdo, a menos que alguien lo haya escrito explícitamente. Carece del trasfondo de conocimiento que cualquier humano da por supuesto.

Esa última fragilidad es justamente la que los modelos de lenguaje de hoy mitigan: traen un sentido común estadístico amplísimo. Pero, como veremos al final, lo hacen a costa de la otra cara de la moneda: pierden la trazabilidad y la garantía que una regla sí ofrece.

Sistemas de soporte a la decisión

Hay un primo cercano de los sistemas expertos que conviene no confundir con ellos: el **sistema de soporte a la decisión** o DSS (del inglés *decision support system*). La diferencia es de filosofía, y es la frase más importante de esta sección: un sistema experto **decide**; un DSS **ayuda a que decida una persona**.¹¹ Un DSS no sustituye el juicio del gestor: le da datos, modelos y escenarios para que su juicio sea mejor.

Un DSS clásico tiene tres componentes, que recuerdan a la arquitectura de un sistema experto pero con otro acento.¹²

COMPONENTE DEL DSS	QUÉ HACE	EQUIVALENTE LEJANO EN EL SISTEMA EXPERTO
Base de datos (gestión de datos)	Reúne datos internos y externos del problema.	Memoria de trabajo y hechos.
Base de modelos (gestión de modelos)	Modelos financieros, estadísticos, de optimización.	Las reglas, pero cuantitativas.
Interfaz de usuario	Deja al humano explorar escenarios «¿y si...?».	El diálogo, pero exploratorio, no concluyente.

La distinción más útil dentro de los DSS es por su componente dominante:

- **Dirigidos por modelos.** El peso está en la base de modelos. El usuario introduce parámetros —tipos de interés, demanda esperada, capacidad— y el modelo calcula consecuencias. Es el simulador financiero que responde «si subimos el precio un 5 %, el margen hace esto y las ventas aquello». Importan pocos datos y un buen modelo.
- **Dirigidos por datos.** El peso está en los datos. El usuario explora grandes volúmenes históricos buscando patrones: cuadros de mando, análisis de ventas por región, segmentación de clientes. Aquí entra de lleno la **inteligencia de negocio** (*business intelligence*), que es la evolución natural de los DSS dirigidos por datos hacia los almacenes de datos, los cubos OLAP y los paneles interactivos.

La relación con la inteligencia de negocio es de continuidad, no de ruptura: el DSS dirigido por datos de los años ochenta y el cuadro de mando moderno responden a la misma pregunta —ayudar a una persona a entender una situación para que decida mejor—, con tecnología de almacenamiento y visualización mucho más potente.

	SISTEMA EXPERTO	SISTEMA DE SOPORTE A LA DECISIÓN
Quién decide	El sistema, aplicando reglas.	La persona, asistida por el sistema.
Qué entrega	Una conclusión y su justificación.	Datos, escenarios, recomendaciones.
Conocimiento	Reglas del experto, cualitativas.	Modelos y datos, cuantitativos.
Mejor para	Tareas estrechas y repetibles.	Decisiones complejas y mal estructuradas.
Responsabilidad	Difusa: ¿responde el sistema?	Clara: responde quien decide.

Esa última fila explica por qué el DSS ha envejecido mejor que el sistema experto autónomo: dejar la decisión —y la responsabilidad— en manos de la persona resultó ser, en casi todos los dominios serios, la arquitectura más sensata. Es la misma lección que SAT, CSP y los

guardrails nos enseñaron en el Facsímil 2: el sistema propone y verifica; la persona, cuando hay algo en juego, dispone.

Híbridos neuro-simbólicos: el regreso de las reglas

Llegamos al presente. Durante años pareció que las redes neuronales habían jubilado a los sistemas de reglas. Hoy el péndulo se ha equilibrado, y el campo que estudia cómo unir las dos tradiciones —el aprendizaje estadístico de las redes y el razonamiento explícito de la lógica— se llama **neuro-simbólico**.¹³ La idea es sencilla y vieja como este capítulo: cada enfoque es fuerte donde el otro es débil.

	RED NEURONAL / LLM	REGLA SIMBÓLICA
Percepción y lenguaje	Excelente.	Pobre.
Sentido común amplio	Sí, estadístico.	No, solo lo escrito.
Garantía de que una regla se cumple	No: es probabilístico.	Sí: se dispara siempre.
Explicación auditable	Difícil.	Trazable por construcción.
Coste de actualizar	Reentrenar.	Editar una regla.

¿Cuándo gana una regla simbólica a un modelo de lenguaje, por muy capaz que sea el modelo? Exactamente cuando necesitas las propiedades de la columna derecha: cuando la decisión tiene que ser **auditable**, cuando una regla debe cumplirse **siempre** y no «casi siempre», y cuando hace falta saber **por qué** se decidió algo. Un permiso es el caso de libro. Si la política dice «un usuario del rol analista no puede aprobar pagos por encima de 50.000 euros», esa frase no puede quedar al criterio de un modelo que el 99,7 % de las veces acierta: el 0,3 % restante es un fraude o una sanción. Quieres una regla que rechace, registre y escale, de forma idéntica, cada vez.

Aquí es donde este capítulo, aparentemente histórico, se enchufa con el resto del facsímil. Los **guardrails** que vimos en el Facsímil 2, capítulo 8, son sistemas de reglas que envuelven a un modelo generativo: validan formato, permisos, coste y estado antes de dejar pasar una salida. Los **permisos de los agentes** del Facsímil 5 son una tabla de reglas que decide qué acciones puede ejecutar un agente y cuáles debe escalar a una persona. En ambos casos, el modelo de lenguaje hace lo que sabe hacer —interpretar, proponer, redactar— y un motor de reglas hace lo que el modelo no garantiza: decidir lo crítico de forma trazable. La arquitectura neuro-simbólica de un agente serio de 2026 es, en su esqueleto, la misma división que MYCIN ya planteaba en 1976: conocimiento explícito por un lado, mecanismo de inferencia por otro, explicación siempre disponible.

La diferencia es que ahora la red neuronal aporta el sentido común que a MYCIN le faltaba, y el motor de reglas aporta la garantía que a una red le falta. Por separado, cada uno tiene un agujero. Juntos, se tapan el agujero mutuamente. Eso es lo que el campo neuro-simbólico persigue, y por eso un capítulo sobre sistemas expertos no es una visita al museo: es la mitad simbólica de la arquitectura que hoy vuelve a construirse.

En el día a día

Sin saberlo, te cruzas con motores de reglas constantemente. El cajero automático que decide si autoriza un reintegro aplica reglas duras —saldo suficiente, tarjeta no bloqueada, dentro del límite diario— antes que cualquier modelo de fraude. El configurador de un coche o un portátil en una web, que te impide elegir una tapicería incompatible con un acabado, es un descendiente directo de XCON. El motor antifraude de tu banco combina un modelo estadístico que puntúa el riesgo con reglas tajantes que bloquean ciertas operaciones pase lo que pase: neuro-simbólico de manual.

El asistente de soporte técnico que te atiende cuando se cae internet es otro motor de reglas con disfraz. Va encadenando hacia atrás desde hipótesis como «el router está bien» o «hay un corte en tu zona», y te pregunta solo lo que necesita para confirmar o descartar cada sospecha —¿parpadea la luz?, ¿has reiniciado?, ¿el cable está enchufado?—, igual que MYCIN pedía solo las pruebas relevantes. El sistema que diagnostica una avería de una lavadora, un ascensor o un motor por sus códigos de error hace lo mismo en sentido contrario, hacia delante: cada código activa una regla que apunta a una causa, y la causa más específica gana a la genérica, como el pingüino le ganaba al ave. Si alguna vez has seguido el árbol de un asistente de averías de un electrodoméstico, has ejecutado a mano un motor de inferencia.

En el desarrollo de software con IA, la conexión es todavía más cercana. Cuando configuras un agente para que pueda leer ficheros pero no borrarlos sin confirmación, estás escribiendo reglas de producción. Cuando un *guardrail* rechaza una respuesta porque contiene un dato personal, es una regla que se ha disparado. El vocabulario ha cambiado —decimos «política», «permiso», «validador»—, pero el mecanismo es el del motor de inferencia que has trazado en este capítulo.

Por qué debería importarte

Porque la tentación de resolverlo todo con un modelo de lenguaje es enorme, y casi siempre equivocada para la parte crítica. Un modelo es magnífico proponiendo, interpretando y redactando, pero no te da una garantía: te da una probabilidad muy alta, que no es lo mismo. Para una decisión auditable, una regla simbólica sigue siendo superior, y entender su mecánica te permite diseñar la frontera correcta: qué dejas al modelo y qué blindas con reglas.

Saber distinguir un sistema que decide de uno que asiste también cambia cómo planteas un producto. Si lo que construyes toma decisiones con consecuencias —dinero, salud, seguridad—, la arquitectura DSS, donde la persona decide y responde, suele ser la elección madura. Confundir las dos cosas es como dejar que el copiloto pilote: cómodo hasta que algo sale mal y nadie sabe quién tenía el mando.

Dónde solía tropezar yo

ERROR	POR QUÉ ES UN ERROR	ANTÍDOTO
Creer que hacia delante y hacia atrás dan resultados distintos	Sobre las mismas reglas y hechos, ambos llegan a las mismas conclusiones; lo que cambia es el camino y el coste.	Elige el sentido por la forma del problema, no por el resultado: datos que llegan → delante; hipótesis que confirmar → atrás.
Meter cientos de reglas sin estrategia de conflicto	Sin refracción entras en bucles; sin prioridad, una regla crítica puede no dispararse a tiempo.	Define la resolución de conflictos antes de crecer: refracción siempre, prioridad explícita para lo crítico.
Reescribir el emparejamiento en cada ciclo	Con muchas reglas, reevaluarlo todo es inviable; el sistema se arrastra.	Usa un motor con Rete (Drools, CLIPS); no reinventes el emparejamiento incremental.
Pedirle a un sistema experto sentido común	Solo sabe lo escrito; fuera de su dominio es ciego y comete absurdos.	Acota el dominio con honestidad y, si necesitas sentido común, añade la mitad neuronal.
Llamar sistema experto a un DSS (o al revés)	Uno decide, el otro asiste; confundirlos difumina quién es responsable.	Pregunta «¿quién firma la decisión?». Si es una persona, es un DSS.
Confiar a un LLM una decisión que debe cumplirse siempre	Un 99,7 % de acierto deja un 0,3 % de fraudes o sanciones.	Blinda lo crítico con una regla que rechaza, registra y escala de forma idéntica.

Cómo encaja todo

Este capítulo es la bisagra entre saber y razonar, y por eso engrana con casi todas las piezas que el facsímil ha ido montando. Conviene recorrer esas conexiones despacio, porque ninguna es de adorno: cada una explica por qué un mecanismo de los años setenta sigue siendo el esqueleto de sistemas que usas todos los días sin verlos.

Empecemos por abajo, por la lógica. En el **capítulo 1 de este mismo facsímil** viste cómo una máquina representa afirmaciones y deriva consecuencias por resolución. Aquello era la gramática; una regla `SI ... ENTONCES ...` no es más que una frase escrita en esa gramática. Cuando el motor comprueba que las premisas de R1 están en la memoria y concluye `empleo_estable`, está haciendo, paso a paso, la misma inferencia que el capítulo 1 demostraba en abstracto. La diferencia es de propósito, no de mecánica: allí se buscaba demostrar un teorema; aquí, imitar el juicio de un analista. Sin la lógica del capítulo 1, las reglas de este serían texto sin garantías; con ella, cada disparo es una deducción válida que se puede defender.

Sube un piso y llegas al conocimiento. El **Facsímil 2, capítulo 12** te entregó la materia prima en estado de reposo: una ontología que dice qué tipos de cosas existen, los hechos que pueblan ese mundo y las reglas que lo gobiernan, modelado con RDF y OWL y consultado con SPARQL. Aquel capítulo describía el saber; este lo pone a funcionar. La TBox y la ABox de las que hablábamos allí son, exactamente, la base de conocimiento de aquí, y el motor de inferencia es lo que las ejecuta. Describir un dominio y razonar sobre él son dos verbos distintos, y la distancia entre ellos es justo la distancia entre aquel capítulo y este: allí dibujaste el mapa; aquí lo recorres.

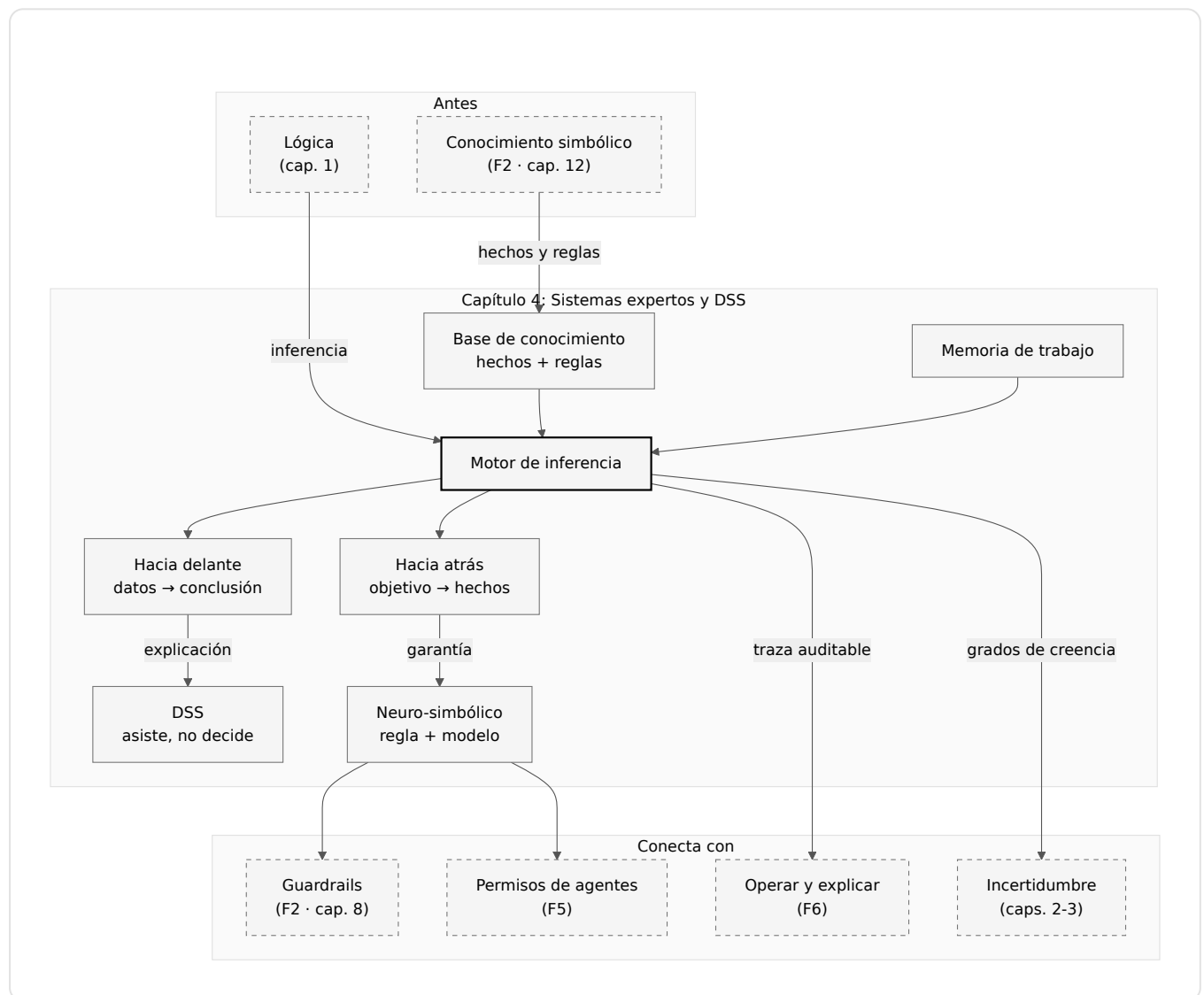
Ahora gira la vista hacia los agentes de hoy, porque es donde todo esto reaparece con ropa nueva. Los **permisos de un agente del Facsímil 5** son, sin más, un pequeño sistema experto. La tabla que decide si una herramienta está autorizada —«el rol analista no puede borrar sin confirmación», «por encima de este importe, escala a una persona»— es una base de reglas, y comprobarla antes de actuar es el ciclo de inferencia que has trazado en este capítulo, reducido a su mínima expresión. Cuando configuras un agente para que lea ficheros pero no los borre, no estás ajustando un parámetro difuso: estás escribiendo reglas de producción que deben dispararse siempre, igual, y poder explicarse. Por eso un permiso nunca debería delegarse en la intuición probabilística del modelo; es, por naturaleza, trabajo de motor de reglas.

Los **guardrails del Facsímil 2, capítulo 8** son el mismo patrón envolviendo a un modelo generativo. Un guardrail que rechaza una respuesta porque contiene un dato personal, o que bloquea una salida mal formada, es una regla que se ha disparado sobre la propuesta del modelo: encadenamiento hacia delante aplicado a lo que el modelo acaba de generar. Y como toda regla, necesita su resolución de conflictos —cuál pesa más cuando varias se activan— exactamente igual que la especificidad y la prioridad que vimos con el pingüino. La frontera correcta de un sistema serio está ahí: el modelo propone e interpreta; el guardrail decide lo innegociable.

Falta la pieza de producción, y es la que cierra el círculo con la trazabilidad. El **Facsímil 6**, sobre operar y explicar decisiones en producción, recoge el testigo del módulo de explicación de este capítulo. Aquella traza de qué reglas se dispararon y por qué, que aquí parecía un detalle de ingeniería, se convierte en lo que un sistema en marcha necesita para responder ante un usuario, un auditor o un incidente: «¿por qué se rechazó este pago?», «¿qué regla bloqueó esta acción?». La explicabilidad que MYCIN ya ofrecía en 1976 no es nostalgia; es justo lo que falta en una red neuronal opaca y lo que un servicio en producción no puede permitirse perder.

Y por encima de todo planea el puente con el presente: lo **neuro-simbólico**. La red neuronal aporta el sentido común y el lenguaje que a los sistemas expertos clásicos les faltaban; el motor de reglas aporta la garantía y la auditabilidad que a una red le faltan. Un agente de 2026, visto por dentro, es la misma división que MYCIN planteaba: conocimiento explícito por un lado, mecanismo de inferencia por otro, explicación siempre disponible, ahora con un modelo de lenguaje ocupando el lugar de la percepción y el trasfondo. Este capítulo, que parecía una visita al museo, es la mitad simbólica de la arquitectura que hoy vuelve a construirse.

Cierra el facsímil mirando hacia delante, dentro de casa. Los **capítulos 2 y 3 de este mismo facsímil**, sobre la incertidumbre, recogen el testigo justo donde MYCIN tropezaba: cuando la evidencia no es un sí o un no rotundos, sino un grado de creencia. Los factores de certeza que MYCIN improvisó para combinar pruebas parciales son el preludio torpe de lo que esos capítulos formalizan con probabilidad. La regla nítida de este capítulo y el grado de creencia de los siguientes son las dos respuestas a la misma pregunta —¿qué hago cuando no lo sé todo?—, y un sistema maduro usa cada una donde encaja.



Vocabulario aprendido

TÉRMINO	DEFINICIÓN
Sistema experto	Programa que resuelve problemas de un dominio aplicando conocimiento experto codificado en reglas.
Base de conocimiento	Conjunto de hechos y reglas que codifican el saber de un dominio.
Memoria de trabajo	Conjunto de hechos conocidos en un momento dado durante el razonamiento.
Motor de inferencia	Mecanismo que aplica las reglas a los hechos para derivar conclusiones.
Encadenamiento hacia delante	Inferencia dirigida por datos: de los hechos a las conclusiones.
Encadenamiento hacia atrás	Inferencia dirigida por objetivos: de la hipótesis a los hechos que la sostienen.
Conjunto de conflicto	Las reglas listas para dispararse en un ciclo del motor.
Resolución de conflictos	Criterio para elegir qué regla disparar cuando varias son aplicables.
Algoritmo Rete	Algoritmo que acelera el emparejamiento de muchas reglas con muchos hechos propagando solo los cambios.
Factor de certeza	Número entre -1 y 1 que mide el grado de creencia en un hecho o una regla; el esquema de MYCIN para arrastrar evidencia parcial por la cadena.
Adquisición del conocimiento	Proceso de extraer y codificar el saber de un experto humano; su gran cuello de botella.
Sistema de soporte a la decisión	Sistema que ayuda a una persona a decidir combinando datos y modelos, sin decidir por ella.
Neuro-simbólico	Enfoque que combina redes neuronales con razonamiento simbólico explícito.

Antes de pasar página

☐ ¿Sé nombrar los componentes de un sistema experto y decir quién escribe cada uno? (Si no, vuelve a «La arquitectura de un sistema experto».)

- ☐ ¿Puedo trazar el encadenamiento hacia delante sobre la base de reglas del préstamo? (Si no, vuelve a «Encadenamiento hacia delante: de los datos a la conclusión».)
- ☐ ¿Entiendo por qué el hacia atrás ignora R5 y R6 y el hacia delante también? (Si no, vuelve a «Encadenamiento hacia atrás: de la hipótesis a los hechos».)
- ☐ ¿Distingo las cuatro estrategias de resolución de conflictos y para qué sirve cada una? (Si no, vuelve a «Resolución de conflictos».)
- ☐ ¿Sé propagar un factor de certeza por una cadena de reglas y decir por qué se desgasta? (Si no, vuelve a «Factores de certeza: arrastrar la incertidumbre por la cadena».)
- ☐ ¿Sé explicar en una frase por qué Rete es eficiente? (Si no, vuelve a «Rete: por qué el emparejamiento no mata el rendimiento».)
- ☐ ¿Puedo decir qué consiguieron MYCIN, DENDRAL y XCON y por qué fueron frágiles? (Si no, vuelve a «Los sistemas expertos clásicos».)
- ☐ ¿Tengo clara la diferencia entre un sistema experto y un DSS? (Si no, vuelve a «Sistemas de soporte a la decisión».)
- ☐ ¿Sé cuándo una regla simbólica gana a un LLM? (Si no, vuelve a «Híbridos neuro-simbólicos».)

En resumen

IDEA FUERZA	DETALLE
Un sistema experto separa qué sabe de cómo razona.	Base de conocimiento y motor de inferencia son piezas independientes.
Hay dos marchas para las mismas reglas.	Hacia delante parte de los datos; hacia atrás, de la hipótesis; llegan a lo mismo por caminos distintos.
La certeza se arrastra por la cadena y se desgasta.	Los factores de certeza de MYCIN combinan premisas con el mínimo y se rebajan a cada regla; encadenar mucho merma la confianza.
El emparejamiento eficiente fue clave.	Rete propaga solo los cambios y reutiliza premisas compartidas.
Los clásicos brillaron y se rompieron por lo mismo.	El conocimiento explícito que les daba potencia era infernal de adquirir y mantener, y carecían de sentido común.
Un DSS asiste; un sistema experto decide.	La diferencia es quién firma la decisión y, por tanto, quién responde.
Las reglas regresan en lo neuro-simbólico.	Cuando una decisión debe ser auditable y cumplirse siempre, una regla gana a un modelo probabilístico.

Inteligencia artificial para gente curiosa · 686f6c61 · CC BY 4.0

Para saber más

Buchanan, B. G. y Shortliffe, E. H. (1984). *Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project*. Addison-Wesley.

Forgy, C. L. (1982). Rete: A fast algorithm for the many pattern/many object pattern match problem. *Artificial Intelligence*, 19(1), 17-37. [https://doi.org/10.1016/0004-3702\(82\)90020-0](https://doi.org/10.1016/0004-3702(82)90020-0)

Garcez, A. S. d'Avila, Lamb, L. C. y Gabbay, D. M. (2009). *Neural-Symbolic Cognitive Reasoning*. Springer.

Giarratano, J. C. y Riley, G. D. (2005). *Expert Systems: Principles and Programming* (4.^a ed.). Thomson Course Technology.

Jackson, P. (1998). *Introduction to Expert Systems* (3.^a ed.). Addison-Wesley.

Power, D. J. (2002). *Decision Support Systems: Concepts and Resources for Managers*. Quorum Books.

Russell, S. y Norvig, P. (2021). *Artificial intelligence: a modern approach* (4.^a ed.). Pearson.

Shortliffe, E. H. (1976). *Computer-Based Medical Consultations: MYCIN*. Elsevier.

Turban, E., Sharda, R. y Delen, D. (2011). *Decision Support and Business Intelligence Systems* (9.^a ed.). Prentice Hall.

Notas

1. Jackson, P. (1998). *Introduction to Expert Systems* (3.^a ed.). Addison-Wesley. El texto define la arquitectura canónica separando base de conocimiento, memoria de trabajo y motor de inferencia. ↵
2. Jackson, P. (1998). *Introduction to Expert Systems* (3.^a ed.). Addison-Wesley. Jackson dedica capítulos enteros a la ingeniería del conocimiento y describe la extracción del saber experto como el obstáculo práctico dominante. ↵
3. Giarratano, J. C. y Riley, G. D. (2005). *Expert Systems: Principles and Programming* (4.^a ed.). Thomson Course Technology. El texto, escrito alrededor del lenguaje CLIPS, desarrolla el encadenamiento hacia delante y el ciclo de reconocimiento-acción en detalle. ↵
4. Russell, S. y Norvig, P. (2021). *Artificial intelligence: a modern approach* (4.^a ed.). Pearson. El tratamiento del razonamiento con reglas presenta el encadenamiento hacia delante y hacia atrás como recorridos complementarios del mismo grafo de inferencia. ↵

5. Shortliffe, E. H. (1976). *Computer-Based Medical Consultations: MYCIN*. Elsevier. El modelo de factores de certeza se presenta como un esquema práctico para combinar evidencia parcial cuando no se dispone de probabilidades fiables. ↵
6. Buchanan, B. G. y Shortliffe, E. H. (1984). *Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project*. Addison-Wesley. La obra discute las limitaciones del modelo de factores de certeza frente a un tratamiento probabilístico completo. ↵
7. Giarratano, J. C. y Riley, G. D. (2005). *Expert Systems: Principles and Programming* (4.ª ed.). Thomson Course Technology. Las estrategias de resolución de conflictos —refracción, novedad, especificidad y prioridad— se describen como parte del ciclo de control de CLIPS. ↵
8. Forgy, C. L. (1982). Rete: A fast algorithm for the many pattern/many object pattern match problem. *Artificial Intelligence*, 19(1), 17-37. [https://doi.org/10.1016/0004-3702\(82\)90020-0](https://doi.org/10.1016/0004-3702(82)90020-0) ↵
9. Shortliffe, E. H. (1976). *Computer-Based Medical Consultations: MYCIN*. Elsevier. La monografía describe la arquitectura de MYCIN, su razonamiento hacia atrás y el modelo de factores de certeza. ↵
10. Buchanan, B. G. y Shortliffe, E. H. (1984). *Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project*. Addison-Wesley. Recoge la experiencia completa de MYCIN, incluidas las dificultades de explicación, adquisición y validación. ↵
11. Power, D. J. (2002). *Decision Support Systems: Concepts and Resources for Managers*. Quorum Books. Power define el DSS por su papel de apoyo al decisor humano y propone una taxonomía por el componente dominante. ↵
12. Turban, E., Sharda, R. y Delen, D. (2011). *Decision Support and Business Intelligence Systems* (9.ª ed.). Prentice Hall. El manual estructura el DSS en gestión de datos, gestión de modelos e interfaz, y sitúa la inteligencia de negocio en continuidad con esta tradición. ↵
13. Garcez, A. S. d'Avila, Lamb, L. C. y Gabbay, D. M. (2009). *Neural-Symbolic Cognitive Reasoning*. Springer. La obra sistematiza la integración de redes neuronales con razonamiento simbólico y argumenta su complementariedad. ↵

Cuadernos para practicar

Escanea el código con el móvil para abrir el cuaderno en Google Colab y ejecutarlo sin instalar nada.



Sistemas expertos: encadenamiento hacia delante y hacia atrás

<https://colab.research.google.com/github/686f6c61/iaparagentecuriosa-notebooks/blob/main/Facs%C3%ADmil%2012/04-encadenamiento-sistemas-expertos.ipynb>

Capítulo 05: Razonamiento causal: de la correlación a la intervención

Entrando en el tema

En el capítulo 2 una máquina aprendió a razonar con incertidumbre. Le dimos grados de creencia, el teorema de Bayes para actualizarlos y las redes bayesianas para organizar muchas creencias a la vez. Con esa maquinaria se responde a una pregunta muy concreta: dado que observo algo, ¿qué debo creer sobre lo demás? La red de la alarma calculaba $P(\text{Robo} \mid \text{Juan, María})$; el clasificador de spam calculaba $P(\text{clase} \mid \text{palabras})$. Todo eran probabilidades condicionales, todo era **observar**.

Pero hay una pregunta que esa maquinaria, por sí sola, no sabe contestar. No es “si veo que la alarma sonó, ¿qué creo del robo?”, sino “si yo **hago** sonar la alarma, ¿cambia algo el robo?”. La respuesta es, obviamente, no: forzar la alarma no provoca ladrones. Y sin embargo, en los datos, alarma y robo están correlacionados. Una probabilidad condicional no distingue las dos preguntas. Las dos se escriben casi igual, pero significan cosas radicalmente distintas, y confundirlas es la fuente de la mayoría de los errores cuando un sistema deja de predecir y empieza a **actuar**.

Este capítulo añade la pieza que faltaba: una semántica de la causa. Vamos a aprender a escribir, calcular y razonar sobre lo que ocurre cuando intervenimos en el mundo, no solo cuando lo miramos. La herramienta es la teoría causal de Judea Pearl, que no tira la probabilidad del capítulo 2 a la basura: la extiende con un operador nuevo, el operador **do**, y con una lectura causal de los mismos grafos dirigidos que ya conocemos.¹ El Facsímil 8, capítulo 7, ya recorrió el lado aplicado de esta historia: experimentos, A/B testing, estimadores como el ATE. Aquí damos el **formalismo** que sostiene aquello: por qué un experimento aleatorizado funciona, qué hace exactamente un ajuste por confusor y qué pregunta responde un contrafactual.

Empecemos por la trampa que justifica todo el capítulo.

La correlación miente: la paradoja de Simpson

Imaginemos un hospital que prueba un fármaco nuevo contra una enfermedad. Recoge datos de 200 pacientes: 100 reciben el fármaco, 100 reciben el tratamiento habitual (control). Mira la tasa de recuperación global y se lleva un disgusto:

GRUPO	PACIENTES	RECUPERAN	TASA
Fármaco nuevo	100	54	54 %
Control	100	76	76 %

Leído sin más, el veredicto es demoledor: el fármaco nuevo **empeora** las cosas, 54 % frente a 76 %. Cualquiera lo retiraría. Pero un médico cauto sospecha que los dos grupos no eran comparables y separa a los pacientes por gravedad de partida, la única variable que faltaba:

GRAVEDAD	TRATAMIENTO	PACIENTES	RECUPERAN	TASA
Leve	Fármaco nuevo	10	9	90 %
Leve	Control	90	72	80 %
Grave	Fármaco nuevo	90	45	50 %
Grave	Control	10	4	40 %

Mira las dos últimas columnas con calma. **Entre los casos leves**, el fármaco gana: 90 % frente a 80 %. **Entre los casos graves**, el fármaco también gana: 50 % frente a 40 %. El fármaco es mejor en cada subgrupo por separado y, a la vez, peor en el total. Esto no es un error de cálculo: las cifras encajan ($9 + 45 = 54$, $72 + 4 = 76$). Es la **paradoja de Simpson**: una tendencia presente en todos los grupos se invierte al juntarlos.²

¿Cómo es posible? Porque el fármaco nuevo se administró sobre todo a los pacientes **graves** (90 de los 100 que lo recibieron), y los graves se recuperan menos hagas lo que hagas. La gravedad es una variable que influye en dos sitios a la vez: en quién recibe el fármaco (los médicos lo reservaban para los casos difíciles) y en quién se recupera. Esa variable contamina la comparación global. El 54 % del fármaco no es bajo porque el fármaco sea malo, sino porque se probó en la población más enferma.

La pregunta del millón es: ¿cuál de las dos lecturas debo creer? ¿El fármaco ayuda o perjudica? Y aquí está la lección central del capítulo: **los datos solos no lo deciden**. Las mismas cifras serían compatibles con conclusiones opuestas según qué historia causal las generó. Si la gravedad es lo que empuja a recetar el fármaco, hay que mirar los subgrupos y el fármaco ayuda. Si fuera al revés —si el fármaco causara la gravedad, cosa absurda aquí pero no siempre—, habría que mirar el total. La estadística no contiene esa información; hay que traerla de fuera, en forma de un modelo de qué causa qué. Eso es exactamente lo que vamos a construir.

La escalera de la causalidad

Pearl ordena las preguntas que podemos hacerle al mundo en tres peldaños, una **escalera de la causalidad** en la que cada nivel necesita más que el anterior y responde preguntas que el anterior no alcanza.³

PELDAÑO	VERBO	PREGUNTA TÍPICA	QUÉ NECESITA
1. Asociación	ver	¿Qué me dice observar X sobre Y ?	Datos y probabilidad condicional.
2. Intervención	hacer	¿Qué pasa con Y si yo fuerzo X ?	Un modelo causal o un experimento.
3. Contrafactual es	imaginar	¿Qué habría pasado con Y si X hubiera sido otra, en este caso concreto?	El modelo causal completo.

El **primer peldaño** es donde vive todo el capítulo 2 y casi todo el aprendizaje automático actual. Se calcula con $P(Y \mid X)$: observo un valor y actualizo mi creencia. Un modelo de este nivel puede ser excelente prediciendo y aun así no saber nada de causas. Sabe que los gallos cantan antes del amanecer; no sabe que silenciarlos no retrasa el sol.

El **segundo peldaño** pregunta por intervenciones. Ya no observo X , lo fijo yo. Se escribe con el operador `do` : $P(Y \mid do(X=x))$. Aquí es donde el ejemplo del fármaco vive de verdad, porque la decisión clínica es una intervención: «si receto este fármaco, ¿se recuperan más?». Ningún volumen de datos puramente observacionales sube por sí solo a este peldaño; hace falta un experimento o un modelo causal que diga qué ajustar.

El **tercer peldaño** es el más alto y el más humano. Pregunta por mundos que no ocurrieron: «este paciente concreto tomó el fármaco y se recuperó; ¿se habría recuperado igualmente sin él?». Es una pregunta sobre un individuo y sobre un pasado alternativo. De ahí salen las nociones de responsabilidad («¿fue su acción la causa del daño?»), de explicación («¿por qué ocurrió?») y de equidad («¿le habrían dado el crédito si hubiera sido de otro grupo?»). La probabilidad condicional ni siquiera sabe formular estas preguntas.

La escalera de la causalidad

Cada peldaño responde preguntas que el anterior no puede ni formular.

más alto en la escalera = más supuestos causales, más poder de respuesta

3 · Imaginar · contrafactuales

¿Qué habría pasado si X hubiera sido otra?

2 · Hacer · intervención

¿Qué pasa con Y si fuerzo X? · $P(Y \mid \text{do}(X))$

1 · Ver · asociación

¿Qué me dice observar X sobre Y? · $P(Y \mid X)$

IA para gente curiosa / Facsímil 12 / Capítulo 05 / 686f6c61

La moraleja operativa es dura: un modelo entrenado solo con datos observacionales vive en el peldaño 1. Si lo usas para decidir una intervención (peldaño 2), estás extrapolando fuera de lo que el modelo realmente sabe, y la paradoja de Simpson es el aviso de lo mal que puede salir. Subir de peldaño no es cuestión de más datos ni de un modelo más grande: es cuestión de añadir información causal que los datos, por su naturaleza, no contienen.

Grafos causales: las flechas significan causas

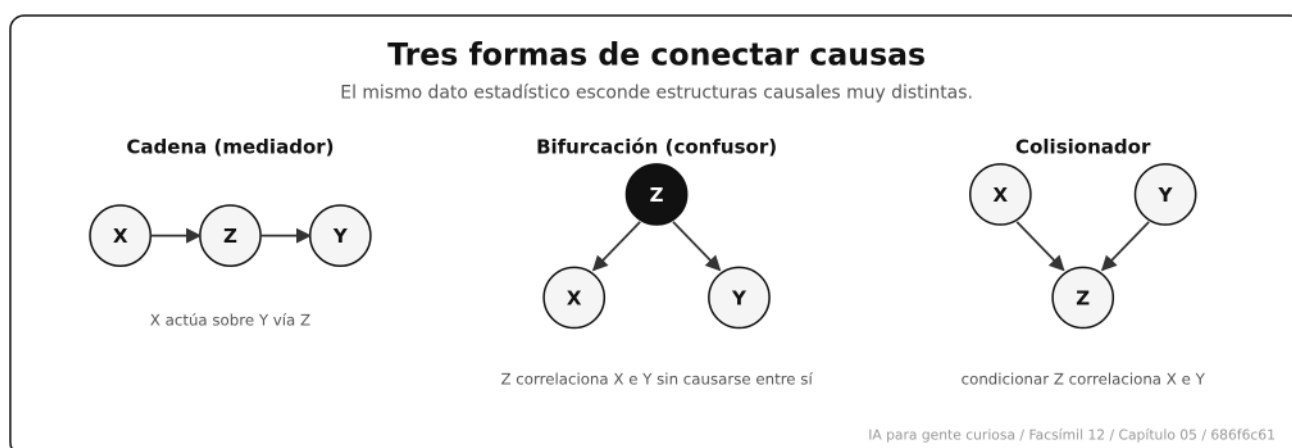
Para subir al segundo peldaño necesitamos escribir nuestras hipótesis sobre qué causa qué. La herramienta es la misma del capítulo 2 con un cambio de lectura: un **grafo dirigido acíclico**, pero ahora cada flecha $X \rightarrow Y$ afirma que X es una **causa directa** de Y , no solo que hay dependencia estadística entre ellas.⁴ Esta es la diferencia profunda con una red bayesiana corriente: una red bayesiana solo promete factorizar bien la distribución conjunta, y la dirección de sus flechas puede elegirse de varias formas equivalentes. Un grafo causal promete algo más fuerte: que si intervienes en un nodo, el efecto se propaga **siguiendo el sentido de las flechas** y nunca hacia atrás.

Como en el capítulo 2, hay tres formas básicas en que tres variables se conectan, y su comportamiento causal es lo que organiza todo lo demás. Conviene tenerlas grabadas, porque decidir bien una intervención es, en el fondo, saber reconocerlas.

PATRÓN	ESTRUCTURA	LECTURA CAUSAL	QUÉ PASA AL CONDICIONAR EL NODO CENTRAL
Cadena	$X \rightarrow Z \rightarrow Y$	X causa Y a través de Z (mediador).	Se bloquea: X e Y se vuelven independientes.
Bifurcación	$X \leftarrow Z \rightarrow Y$	Z es causa común de X e Y (confusor).	Se bloquea: desaparece la correlación espuria.
Colisionador	$X \rightarrow Z \leftarrow Y$	X e Y son causas de Z .	Se abre: aparece correlación falsa entre X e Y .

La **bifurcación** es la villana de este capítulo. Cuando una variable Z causa a la vez a X y a Y , se la llama **confusor**, y es la responsable de la paradoja de Simpson: la gravedad causaba tanto recibir el fármaco como recuperarse. Una bifurcación produce correlación entre X e Y **sin que exista ninguna flecha de X a Y** . Helados y ahogamientos suben juntos no porque el helado ahogue, sino porque el calor (la Z) provoca ambas cosas. Quien confunde esa correlación con causa actuaría prohibiendo los helados para salvar vidas.

El **colisionador** es la trampa contraria y la más sutil. Aquí X e Y son independientes de partida, pero condicionar sobre su efecto común Z las correlaciona artificialmente. Es el *explaining away* del capítulo 2 leído en clave causal: si dos causas independientes pueden producir un mismo efecto y observamos el efecto, saber que una causa estuvo presente reduce la sospecha de la otra. El peligro práctico es feroz: ajustar (condicionar) por un colisionador, lejos de limpiar la comparación, la **ensucia**. No todo lo que se puede medir se debe controlar; controlar la variable equivocada introduce sesgo en lugar de quitarlo.



El operador do: observar no es intervenir

Volvamos al fármaco y escribamos las dos preguntas que tanto se parecen. El grafo causal del experimento es una bifurcación: la gravedad Z causa el tratamiento X (los médicos recetaban según la gravedad) y causa la recuperación Y ; y el tratamiento puede causar la recuperación. Es decir, $Z \rightarrow X$, $Z \rightarrow Y$ y $X \rightarrow Y$.

La pregunta de **observación** es $P(Y \mid X=x)$: entre los pacientes que de hecho tomaron el fármaco, ¿cuántos se recuperaron? Esa cifra mezcla dos cosas: el efecto real del fármaco y el hecho de que el fármaco fue a parar a los pacientes más graves. Es el 54 % engañoso.

La pregunta de **intervención** es $P(Y \mid do(X=x))$: si yo administrara el fármaco a un paciente **sin que la gravedad influya en esa decisión**, ¿cuántos se recuperarían? El operador `do` representa precisamente eso: agarrar la variable X y fijarla por la fuerza, ignorando sus causas habituales. Gráficamente, intervenir con `do(X=x)` equivale a **borrar todas las flechas que entran en X** : la gravedad ya no decide el tratamiento, lo decido yo. Esa amputación quirúrgica del grafo es la definición de Pearl, y es lo que un experimento aleatorizado consigue físicamente: al echar a suertes quién recibe el fármaco, rompemos el vínculo entre la gravedad y el tratamiento. Por eso la aleatorización es la regla de oro, y por eso el Facsímil 8, capítulo 7, dedica tanto cuidado a que la asignación esté de verdad rota respecto a todo lo demás.

$$P(Y \mid X=x) \neq P(Y \mid do(X=x))$$

observar: el confusor sigue actuando

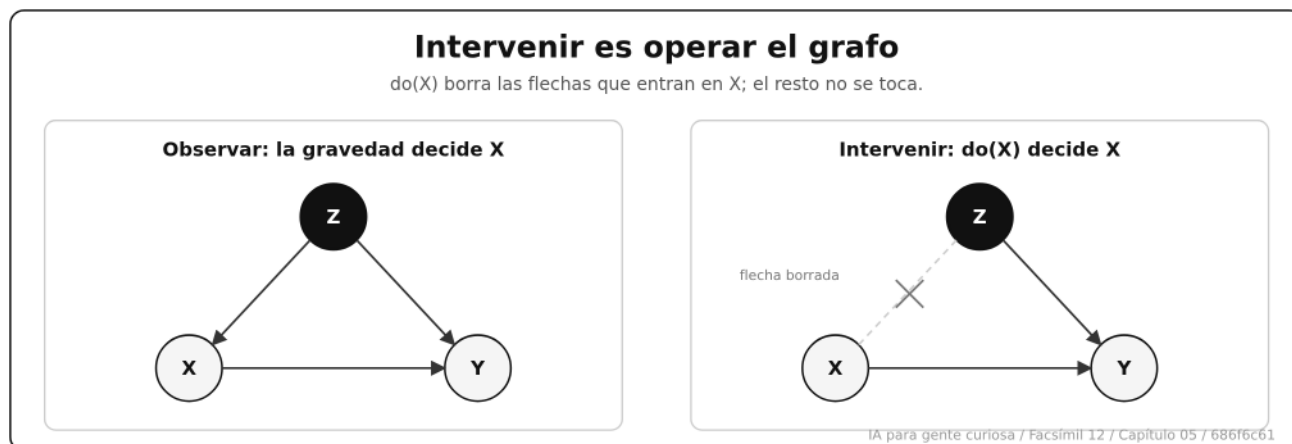
$$\neq$$

intervenir: el confusor ya no decide X

SÍMBOLO	SIGNIFICADO	EJEMPLO
$P(Y \mid X=x)$	Probabilidad de Y entre quienes se observa que tienen $X=x$.	Recuperación entre los que de hecho tomaron el fármaco (el 54 %).
$P(Y \mid do(X=x))$	Probabilidad de Y si fijamos $X=x$ por intervención.	Recuperación si administramos el fármaco al margen de la gravedad (el 70 %).
$do(X=x)$	Operador de intervención: borra las flechas que entran en X .	Repartir el fármaco a suertes, no según el criterio del médico.
$\neq$	Los dos lados difieren mientras quede un confusor sin cerrar.	54 % observado frente a 70 % intervenido.

En palabras: lo que ves entre quienes ya tienen un valor de la causa **no** es lo mismo que lo que pasaría si tú impusieras ese valor a todos. La izquierda mezcla el efecto de la causa con la huella de quién acabó teniéndola; la derecha aísla el efecto, porque al intervenir el confusor deja de elegir quién recibe el tratamiento. Solo coinciden cuando no hay nada que confunda.

Esa amputación de las flechas que entran en X es literal en el grafo. A la izquierda, el mundo observado: la gravedad decide el tratamiento y el resultado a la vez. A la derecha, el mundo intervenido con $do(X=x)$: la flecha de la gravedad al tratamiento desaparece, porque ahora quien decide el tratamiento eres tú, no la gravedad. El resto del grafo se queda intacto.



La diferencia entre los dos lados de esa desigualdad es el **sesgo del confusor**. Cuando no hay confusores (cuando ninguna variable causa a la vez X e Y), los dos lados coinciden y observar equivale a intervenir; ese es el mundo cómodo de un experimento bien aleatorizado. Cuando sí los hay, como en el fármaco, la observación está sesgada y hay que corregirla. La pregunta es cómo, y la respuesta tiene nombre.

El criterio de la puerta trasera y la fórmula de ajuste

El sesgo del confusor entra por un camino concreto del grafo. Entre el tratamiento X y el resultado Y hay dos tipos de caminos: el **camino causal** que va de X hacia Y siguiendo las flechas ($X \rightarrow Y$), que es el que queremos medir, y los **caminos de puerta trasera**, que conectan X e Y pero empiezan con una flecha que entra en X . En el fármaco, ese camino traicionero es $X \leftarrow Z \rightarrow Y$: sale de X por detrás, pasa por la gravedad y desemboca en la recuperación. Por ahí se cuela la correlación espuria.

El **criterio de la puerta trasera** de Pearl dice cómo cerrar esos caminos: hay que encontrar un conjunto de variables Z que, al condicionar sobre ellas, **bloquee todos los caminos de puerta trasera** sin bloquear el camino causal y sin abrir colisionadores. Cuando ese conjunto existe y lo observamos, el efecto causal se vuelve calculable a partir de datos observacionales mediante la **fórmula de ajuste**:

$$P(Y \mid do(X=x)) = \sum_z P(Y \mid X=x, Z=z) P(Z=z)$$

SÍMBOLO	SIGNIFICADO	EJEMPLO
$P(Y \mid do(X=x))$	Probabilidad de Y si intervenimos fijando $X=x$.	Recuperación si administramos el fármaco.
$X=x$	La causa, fijada al valor que evaluamos.	fármaco = sí .
$Z=z$	Cada valor del confusor que ajustamos.	Gravedad leve o grave.
$P(Y \mid X=x, Z=z)$	Tasa observada de Y dentro de cada subgrupo.	Recuperación de tratados leves, de tratados graves.
$P(Z=z)$	Peso de cada subgrupo en la población entera .	Qué fracción de pacientes es leve y cuál grave.
$\sum_z$	Suma sobre todos los valores del confusor.	Recombinar los subgrupos.

En palabras: mide el efecto del tratamiento **dentro de cada subgrupo del confusor**, donde la comparación sí es justa, y luego promedia esos efectos usando como pesos lo que pesa cada subgrupo en la población total, no en el grupo tratado. Esa última coletilla es la clave: el confusor sesgaba porque el tratamiento se repartía mal entre subgrupos; al promediar con los pesos de la población completa, deshacemos ese reparto torcido.

Hagamos el cálculo con las cifras del hospital. De los 200 pacientes, 100 son leves y 100 graves, así que $P(Z=\text{leve}) = P(Z=\text{grave}) = 0,5$. Las tasas de recuperación por subgrupo ya las tenemos en la tabla. Para el fármaco:

$$P(Y \mid do(\text{fármaco})) = \underset{\text{leves}}{0,90} \cdot 0,5 + \underset{\text{graves}}{0,50} \cdot 0,5 = 0,45 + 0,25 = 0,70$$

Y para el control, repitiendo la operación con sus tasas:

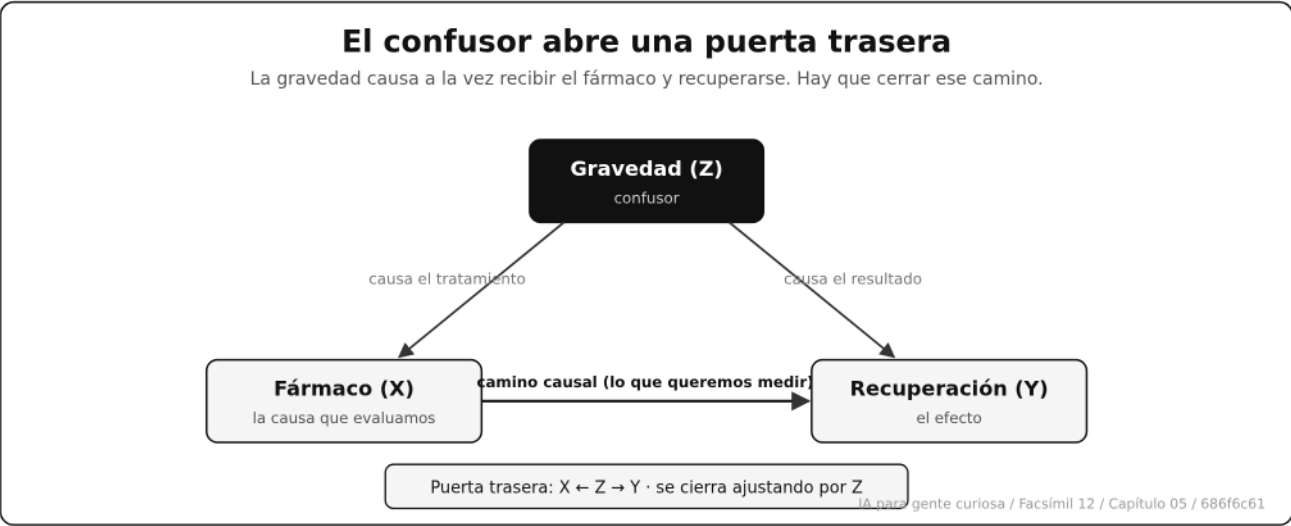
$$P(Y \mid do(\text{control})) = 0,80 \cdot 0,5 + 0,40 \cdot 0,5 = 0,40 + 0,20 = 0,60$$

En palabras: en lugar de comparar el grupo tratado con el de control tal como quedaron (uno lleno de graves, el otro de leves), cogemos la tasa de cada subgrupo y la pesamos por lo que ese subgrupo representa en **toda** la población, mitad y mitad. Al usar los mismos pesos para los dos brazos, ambos se evalúan sobre una población idéntica y la comparación deja de estar contaminada por el reparto torcido del tratamiento.

El efecto causal del fármaco es $0,70 - 0,60 = +0,10$: **diez puntos porcentuales más de recuperación**. El fármaco ayuda, justo lo contrario de lo que gritaba la cifra global. Compara las dos lecturas en una tabla:

CANTIDAD	CÁLCULO	RESULTADO	CONCLUSIÓN
Observación ingenua	$P(Y \mid \text{fármaco}) - P(Y \mid \text{control})$	$0,54 - 0,76 = -0,22$	El fármaco parece dañino.
Intervención ajustada	$P(Y \mid \text{do}(\text{fármaco})) - P(Y \mid \text{do}(\text{control}))$	$0,70 - 0,60 = +0,10$	El fármaco ayuda.

El ajuste por el confusor **invierte la conclusión**, de -22 puntos a $+10$. Y esto no es magia estadística: es la consecuencia matemática de haber escrito el grafo causal correcto y de haber bloqueado el camino de puerta trasera $X \leftarrow Z \rightarrow Y$. Si la gravedad no hubiera sido un confusor, ajustar por ella no habría cambiado nada. El grafo es lo que nos dice **qué ajustar**; sin él, ajustar por una variable es tan probable que ayude como que estropee.



Conviene subrayar lo que la fórmula de ajuste **no** es. No es «controlar por todas las variables que tengas». Recuerda el colisionador: ajustar por la variable equivocada introduce sesgo. El criterio de la puerta trasera es preciso justamente para evitar ese error: ajusta por lo que cierra puertas traseras, deja en paz mediadores y colisionadores. Y tiene un límite honesto: solo funciona si el confusor está **medido**. Si la gravedad no se hubiera registrado, ningún cálculo la habría recuperado; haría falta un experimento. Esa es la diferencia entre tener datos y poder concluir causas.

El colisionador: cuando condicionar inventa correlación

Si el confusor es la trampa que nos empuja a ajustar, el **colisionador** es la trampa que nos castiga por ajustar de más. Su criterio es la imagen invertida del de la puerta trasera, y conviene enunciarlo aparte porque es responsable de algunos de los errores más caros y menos evidentes del análisis de datos: **condicionar sobre un colisionador (o sobre**

cualquier descendiente suyo) abre un camino que estaba cerrado y crea correlación donde no había ninguna. No hace falta ninguna flecha entre X e Y para que, tras filtrar por su efecto común, parezcan relacionados.

El mecanismo es el *explaining away* del capítulo 2 visto en otra clave. Tomemos un ejemplo de calle. Imagina dos rasgos de los actores que llegan a ser famosos: el **talento** X y el **atractivo físico** Y . En la población general no tienen por qué ir juntos: hay gente con mucho talento y poco atractivo, y al revés. Pero para hacerse famoso ayuda cualquiera de los dos, así que la **fama** Z es un colisionador, $X \rightarrow Z \leftarrow Y$. Ahora haz lo que hace cualquiera sin darse cuenta: mira **solo a los famosos**. Dentro de ese grupo aparece una correlación negativa entre talento y atractivo que en la población entera no existía. ¿Por qué? Porque si alguien es famoso y resulta que tiene poco talento, algo lo explica: seguramente es muy atractivo. Filtrar por el efecto común hace que sus dos causas compitan por explicarlo, y eso las correlaciona artificialmente. Quien observara solo a los famosos concluiría, falsamente, que el talento «estropea» el atractivo.

El mismo patrón, con otro disfraz, aparece por todas partes:

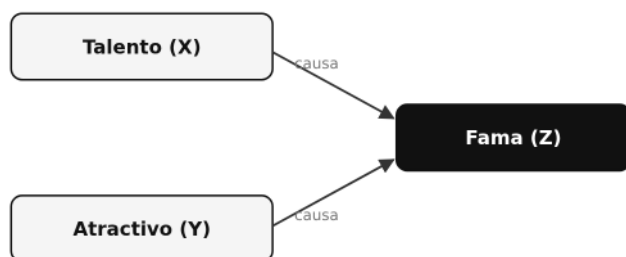
- **Admisiones y becas.** Una universidad admite por nota o por mérito deportivo. Entre los admitidos, ambas cosas salen correlacionadas a la baja, aunque en los aspirantes fueran independientes. Estudiar solo a los admitidos engaña.
- **Sesgo de selección en sanidad.** Si analizas únicamente a pacientes **hospitalizados**, dos enfermedades que llevan al hospital pueden parecer asociadas (la *paradoja de Berkson*) sin que una tenga nada que ver con la otra.
- **Contratación.** Entre los empleados ya contratados, la destreza técnica y la soltura en la entrevista pueden correlacionar negativamente: si te contrataron pese a una entrevista floja, tu perfil técnico debía de ser muy fuerte.

La regla práctica es incómoda porque contradice el instinto de «cuantas más variables controle, más limpio el análisis»: **no todo lo que se puede medir se debe controlar.** Condicionar (ajustar, estratificar, o simplemente seleccionar la muestra) por un colisionador no quita sesgo, lo fabrica. El grafo causal es, de nuevo, lo único que distingue un confusor (que sí hay que ajustar) de un colisionador (que hay que dejar en paz), porque en los datos brutos ambos se ven igual de correlacionados.

Condicionar sobre un colisionador inventa correlación

Talento y atractivo son independientes, salvo que mires solo a los famosos.

Población entera: sin relación



Solo famosos: correlación falsa



IA para gente curiosa / Facsímil 12 / Capítulo 05 / 686f6c61

Hay un tercer error de ajuste, simétrico del anterior, que conviene nombrar para cerrar el cuadro: ajustar por un **mediador**. Si la causa actúa **a través** de una variable intermedia, $X \rightarrow M \rightarrow Y$, esa M no es un confusor ni un colisionador, sino el propio canal por el que viaja el efecto. Imagina que el fármaco cura reduciendo la inflamación, y que la inflamación reducida es lo que mejora la recuperación. Si controlas por la inflamación, **borras** justo la vía por la que el fármaco hace su trabajo, y el efecto medido se desploma hacia cero aunque el fármaco funcione. Esto se llama **sesgo de sobrecontrol**, y es la cara opuesta del colisionador: el colisionador inventa correlación, el mediador la suprime. La conclusión une las tres figuras en una sola regla: ajusta por confusores, deja en paz mediadores y colisionadores. Cuál es cuál no lo dicen los datos, lo dice el grafo.

Contrafactuales: qué habría pasado si

El tercer peldaño pregunta por mundos alternativos sobre casos concretos. «María tomó el fármaco y se recuperó. ¿Se habría recuperado igualmente sin tomarlo?» Esta no es una pregunta sobre promedios de población, sino sobre **un individuo y un pasado que no ocurrió**. Si la respuesta es «sí, se habría recuperado igual», entonces el fármaco no fue la causa de su recuperación, aunque la tomara y se curara. Si la respuesta es «no, sin el fármaco no se habría recuperado», entonces el fármaco fue, para ella, decisivo.

Formalizar contrafactuales requiere maquinaria que va más allá de lo que cubre este capítulo: hace falta el modelo causal completo, con sus ecuaciones, para «rebobinar» el caso, cambiar el valor de la causa y volver a simular el desenlace. Pearl describe ese rebobinado en tres pasos, y aunque la mecánica completa quede fuera de aquí, el esquema mental es sencillo y vale la pena tenerlo:

1. **Mirar el caso real y deducir lo que no se ve.** María tomó el fármaco y se recuperó. De ese hecho, y del modelo, inferimos las circunstancias individuales que el promedio de la población esconde: su gravedad, su respuesta al tratamiento, todo lo que la hace ser ella y no «un paciente medio».

2. **Cambiar solo la causa, dejando lo demás fijo.** Imaginamos a esa misma María, con esas mismas circunstancias, pero sin el fármaco. No es otra paciente parecida: es ella, con un único cambio.
3. **Volver a calcular el desenlace.** Con la causa cambiada y todo lo demás congelado, preguntamos si se habría recuperado.

Pongamos números a la intuición. Supongamos que el modelo dice que, en su subgrupo, el fármaco solo es decisivo para una parte de los pacientes; para el resto, se habrían curado igual o no se habrían curado de ninguna manera. Si las circunstancias de María la sitúan entre los primeros, el contrafactual «sin fármaco, no se recupera» es verdadero **para ella**, aunque a escala de población el fármaco solo suba la tasa diez puntos. Aquí está la diferencia crucial: el efecto promedio de la intervención (peldaño 2) responde «de cada cien tratados, diez se curan que no se habrían curado»; el contrafactual (peldaño 3) responde «**esta** persona concreta, ¿es una de esas diez?». Son preguntas distintas, y la segunda exige más supuestos que la primera, porque mezcla lo observado con un pasado que no ocurrió.

La intuición de los dos mundos posibles para un mismo caso (lo que pasa con tratamiento y lo que pasaría sin él) la formalizó Rubin con los resultados potenciales, el lenguaje hermano del de Pearl para hablar de efectos individuales que solo observamos a medias.⁵ De cada persona solo vemos uno de los dos mundos (tomó el fármaco o no lo tomó), nunca los dos a la vez; ese es el **problema fundamental de la inferencia causal**, y es la razón de que un contrafactual individual nunca se compruebe directamente, solo se estime con un modelo. Pero la **idea** es accesible y es la que importa para entender por qué la causalidad es indispensable. Tres lugares donde un contrafactual hace todo el trabajo:

- **Responsabilidad.** Decir que una acción «causó» un daño es, en el fondo, afirmar un contrafactual: sin esa acción, el daño no habría ocurrido. Los tribunales razonan así («but for» en el derecho anglosajón). Un sistema de IA que reparte culpas o méritos está, lo sepa o no, haciendo afirmaciones contrafactuales.
- **Equidad.** Una decisión es injusta por discriminación si **habría sido distinta** caso de cambiar solo el atributo protegido (género, etnia) manteniendo todo lo demás. Eso es un contrafactual puro, y es la base de las definiciones causales de equidad que veremos asomar en el Facsímil 9.
- **Explicación.** «¿Por qué me denegaron el crédito?» se responde mejor con un contrafactual accionable: «si tus ingresos hubieran sido 3 000 en lugar de 2 000, te lo habrían concedido». Eso le dice a la persona qué cambiar; una mera correlación, no.

El caso de la equidad merece un segundo vistazo porque enlaza directamente con el Facsímil 9. Un modelo puede no usar el género como entrada y aun así discriminar, si se apoya en una variable que **es** consecuencia del género (el tipo de empleo, el historial, el barrio). Distinguir cuándo eso es injusto exige un contrafactual: «¿la decisión sería la misma si esta persona hubiera nacido de otro género, arrastrando los cambios que ello implica?». Esa pregunta es la que separa una correlación legítima con el resultado de un camino de discriminación, y es justo el tipo de distinción que una métrica puramente estadística no sabe hacer; por eso muchas definiciones de equidad numéricas se contradicen entre sí. Aquí también acecha el

colisionador: condicionar sobre una variable de selección (quién fue contratado, quién fue admitido) puede inventar o esconder disparidades entre grupos, y solo el grafo causal dice si la cifra que miras es real o un artefacto de a quién estás mirando.

La conexión con el capítulo 2 es elegante: una red bayesiana corriente vive en el peldaño 1, un grafo causal con el operador `do` sube al peldaño 2, y un modelo causal estructural completo alcanza el peldaño 3. Es la misma familia de grafos ganando, en cada nivel, más capacidad de respuesta a cambio de más compromiso con cómo funciona el mundo.⁶

En el día a día

Aunque suene a manual, esta distinción decide cosas todos los días. Un panel de marketing muestra que los usuarios que reciben la newsletter compran más; ¿conviene mandarla a todos? Solo si el efecto es causal y no un confusor (quizá quien se suscribe ya era más fiel). Un hospital ve que los pacientes ingresados en cierta unidad mueren más; ¿es peor la unidad, o es que allí mandan los casos más graves? Es la paradoja de Simpson otra vez, con vidas de por medio. Un equipo de producto observa que los usuarios que usan una función avanzada retienen mejor; ¿forzar la función mejorará la retención, o solo los usuarios ya comprometidos la descubren?

La misma trampa asoma en la calle comercial. **Precios y demanda:** los datos muestran que los meses en que más coches vende un concesionario son los de precio más alto; ¿subir el precio venderá más? Por supuesto que no: el confusor es la temporada (en primavera sube la demanda y suben los precios a la vez). Quien lea esa correlación como una palanca se arruina. **Publicidad y ventas:** una marca gasta más en anuncios justo en campaña de rebajas, cuando las ventas suben por sí solas; el panel atribuye al anuncio un efecto que era del calendario. **Ejercicio y salud:** quien hace deporte vive más, sí, pero parte de esa correlación la sostiene un confusor (quien ya está sano hace más deporte), y solo un diseño cuidadoso separa el efecto real del ejercicio de la salud previa que lo permite. En todos, la pregunta es la misma: ¿la cifra mide el efecto de mover la palanca, o solo refleja quién acabó moviéndola?

Por eso el patrón oro sigue siendo intervenir de verdad. El **A/B testing** es exactamente el operador `do` hecho producto: al repartir a los usuarios a suertes entre la versión A y la B, rompemos por diseño cualquier vínculo entre la asignación y los rasgos del usuario, de modo que la diferencia observada **sí** es $P(Y \mid do(X))$ y no una correlación contaminada. Cuando no se puede experimentar (por coste, ética o tiempo), queda la otra vía: ajustar por los confusores correctos guiados por un grafo causal. Lo que nunca es válido es tratar una correlación observada como si fuera el resultado de una intervención.

La pregunta higiénica que conviene hacerse siempre es: «¿esta cifra viene de **mirar** o de **hacer**?». Si viene de mirar y vas a actuar, falta un paso. Ese paso es, según los casos, un experimento (Facsimil 8, capítulo 7) o un ajuste por los confusores correctos guiado por un grafo causal.

Por qué debería importarte

Porque la IA moderna está, casi toda, atrapada en el primer peldaño. Un modelo entrenado por máxima verosimilitud aprende correlaciones: $P(Y | X)$ a gran escala.⁷ Eso basta para predecir, y a veces deslumbra. Pero en cuanto el modelo **actúa** (recomienda, prioriza, decide a quién se concede un crédito, qué prueba se ordena, qué caso se escala), está tomando decisiones de intervención con una herramienta de observación. Y ahí aparecen tres fallos caros.

El primero son las **correlaciones espurias y los atajos** (*shortcut learning*): un clasificador que distingue lobos de huskies fijándose en la nieve del fondo, o un modelo médico que detecta neumonía por el tipo de aparato de rayos X en lugar de por la patología. Funcionan en los datos de entrenamiento porque la correlación está ahí, y fallan estrepitosamente cuando el mundo cambia o cuando intervienes. Son confusores no controlados disfrazados de rasgos predictivos. Visto con la lente de este capítulo, el atajo es exactamente el camino de puerta trasera del clasificador: en el aparato de rayos X concreto influye el tipo de hospital, que a la vez se asocia con la gravedad de los pacientes que recibe; el modelo aprende ese rodeo en lugar del camino causal que va de la patología al diagnóstico. Por eso un atajo no se arregla con más datos del mismo sitio, igual que el sesgo del confusor no se arreglaba con más filas: lo que falta es saber qué rasgo es causa y cuál es solo compañía de viaje.

El segundo es la **falta de robustez**. Un modelo que solo correlaciona se hunde cuando la distribución se mueve, porque las correlaciones espurias no sobreviven al cambio de entorno mientras que las relaciones causales sí tienden a hacerlo. Aprender la causa, y no el atajo, es una de las apuestas más serias para construir IA que no se rompa fuera del laboratorio.

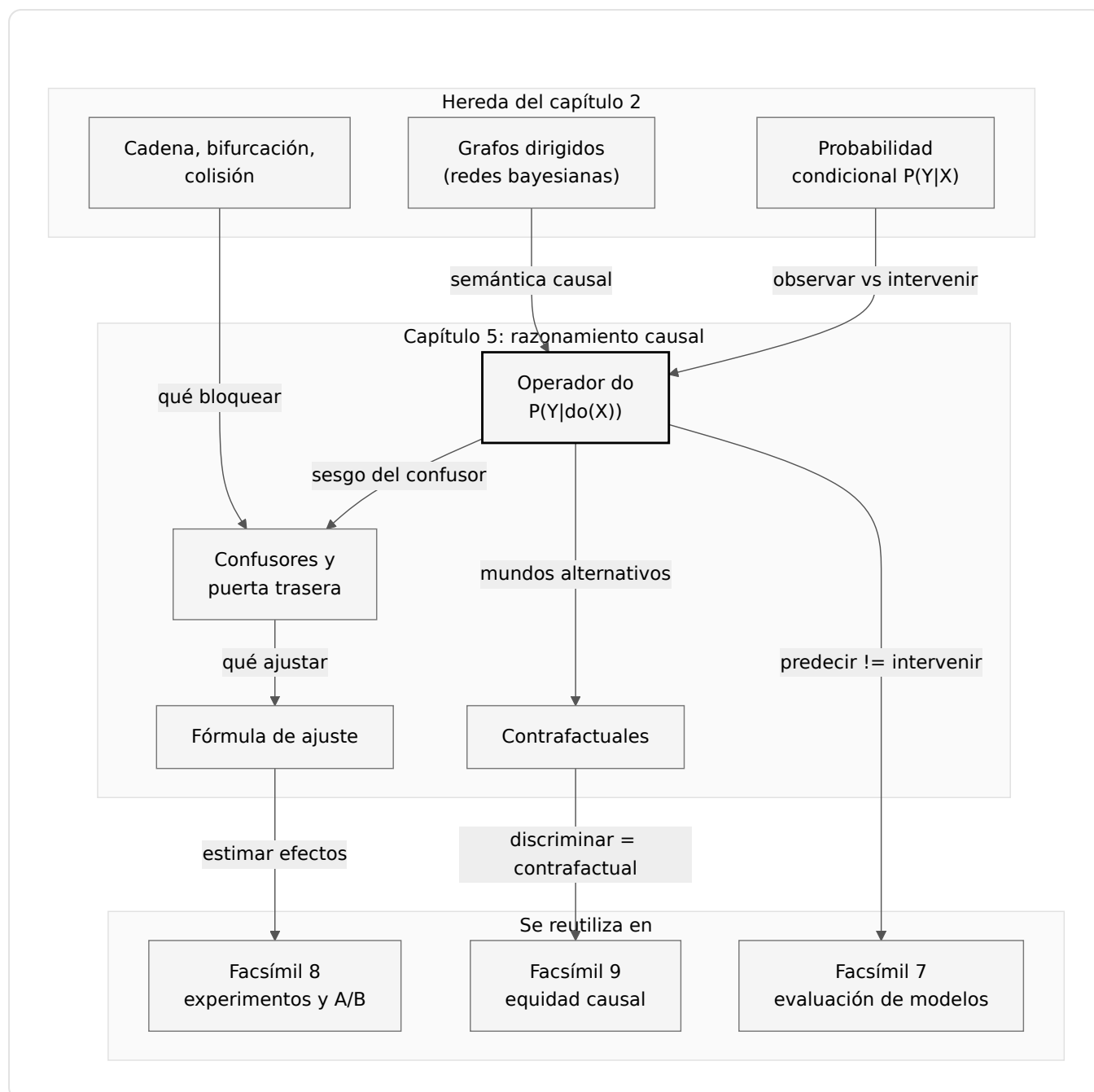
El tercero es la **equidad**. Como vimos, discriminar es un concepto contrafactual, y muchas métricas de equidad puramente estadísticas se contradicen entre sí precisamente porque ignoran la estructura causal de cómo se generan los datos. El Facsímil 9 retoma este hilo: sin causalidad, «justo» es una palabra que cada cual rellena a su gusto. La causalidad no resuelve sola el problema ético, pero da el lenguaje para plantearlo sin trampas.

Dónde solía tropezar yo

ERROR	POR QUÉ ES UN ERROR	ANTÍDOTO
Leer $P(Y X)$ como si fuera $P(Y do(X))$	Observar e intervenir coinciden solo si no hay confusores; con confusores, la observación está sesgada.	Pregúntate si la cifra viene de mirar o de hacer. Si vas a actuar, ajusta o experimenta.
Ajustar por todo lo que tengo medido	Condicionar sobre un colisionador o un mediador introduce sesgo en lugar de quitarlo.	Dibuja el grafo y aplica el criterio de la puerta trasera: ajusta solo lo que cierra puertas traseras.
Creer que más datos resuelven la confusión	El sesgo del confusor no desaparece con el tamaño de la muestra; un millón de filas confundidas siguen confundidas.	Lo que falta es información causal (qué causa qué), no más filas.
Mirar solo la cifra agregada	La paradoja de Simpson invierte conclusiones al juntar grupos con distinto reparto del tratamiento.	Desagrega por el confusor sospechoso y compara con el agregado antes de concluir.
Confundir un buen predictor con una buena palanca	Una variable puede predecir Y sin causarla; forzarla no mueve Y .	Antes de intervenir sobre una variable, comprueba que hay una flecha causal hacia el resultado, no solo correlación.

Cómo encaja todo

Este capítulo cierra el facsímil dando el salto que faltaba: de la creencia a la causa. Hereda directamente del capítulo 2 los grafos dirigidos y la probabilidad condicional, pero les añade una semántica nueva (el operador `do`) que distingue observar de intervenir. Lo que aquí se aprende (grafos causales, confusores, puerta trasera, ajuste y contrafactuales) es el cimiento formal de tres lugares del facsímil: los experimentos y la causalidad aplicada del Facsímil 8, las definiciones causales de equidad del Facsímil 9, y la evaluación honesta de modelos del Facsímil 7, que solo tiene sentido si distinguimos lo que un modelo predice de lo que provoca al actuar.



Leído en palabras: las redes bayesianas del capítulo 2 nos dieron los grafos y la probabilidad condicional, pero se quedaban en el peldaño de la asociación. Al cambiar la lectura de las flechas a «causa directa» y añadir el operador **do**, subimos al peldaño de la intervención; los patrones de conexión que ya conocíamos (cadena, bifurcación, colisión) nos dicen ahora qué variables bloquear, el criterio de la puerta trasera elige el conjunto correcto y la fórmula de ajuste lo convierte en un cálculo. De ahí salen, hacia delante, la estimación de efectos del Facsímil 8, las definiciones contrafactuales de equidad del Facsímil 9 y la disciplina del Facsímil 7 de no confundir un buen predictor con una buena palanca.

Vocabulario aprendido

TÉRMINO	DEFINICIÓN
Correlación frente a causalidad	Que dos cosas varíen juntas no implica que una cause la otra.
Grafo causal	Grafo dirigido acíclico cuyas flechas representan relaciones de causa a efecto.
Operador do	Notación de Pearl para una intervención: fijar una variable por la fuerza, no observarla.
Confusor	Variable que causa a la vez la supuesta causa y el efecto, y crea correlación espuria.
Criterio de la puerta trasera	Regla gráfica para elegir qué variables ajustar y estimar un efecto causal.
Fórmula de ajuste	Cálculo del efecto de una intervención sumando sobre los confusores controlados.
Colisionador	Variable causada por otras dos; condicionar sobre ella crea correlación falsa entre sus causas.
Escalera de la causalidad	Los tres niveles de Pearl: asociación (ver), intervención (hacer) y contrafactuales (imaginar).
Contrafactual	Afirmación sobre qué habría pasado si algo hubiera sido distinto.
Paradoja de Simpson	Una tendencia que aparece en grupos separados se invierte al juntarlos.

Antes de pasar página

- ☐ ¿Sé explicar con un ejemplo por qué $P(Y | X)$ y $P(Y | do(X))$ son distintas? (Si no, vuelve a «El operador do: observar no es intervenir».)
- ☐ ¿Entiendo cómo la paradoja de Simpson invierte una conclusión y qué papel juega el confusor? (Si no, vuelve a «La correlación miente: la paradoja de Simpson».)
- ☐ ¿Distingo los tres peldaños de la escalera de la causalidad y qué pregunta responde cada uno? (Si no, vuelve a «La escalera de la causalidad».)
- ☐ ¿Reconozco una cadena, una bifurcación y un colisionador, y sé qué pasa al condicionar el nodo central de cada uno? (Si no, vuelve a «Grafos causales: las flechas significan causas».)

- ☐ ¿Sé aplicar la fórmula de ajuste por puerta trasera y por qué uso los pesos de la población entera? (Si no, vuelve a «El criterio de la puerta trasera y la fórmula de ajuste».)
- ☐ ¿Entiendo por qué ajustar (o seleccionar la muestra) por un colisionador mete sesgo en lugar de quitarlo? (Si no, vuelve a «El colisionador: cuando condicionar inventa correlación».)
- ☐ ¿Sé formular un contrafactual y conectarlo con responsabilidad, equidad o explicación? (Si no, vuelve a «Contrafactuales: qué habría pasado si».)
- ☐ ¿Sé decir qué lectura (efecto ingenuo o ajustado) usaría para decidir y por qué? (Si no, repasa la sección correspondiente del capítulo.)

En resumen

IDEA FUERZA	DETALLE
La correlación no basta para decidir.	$P(Y X)$ responde qué observo; para actuar necesito $P(Y do(X))$, que puede ser muy distinta.
La paradoja de Simpson es el aviso.	Una tendencia presente en cada subgrupo puede invertirse al agregar, por culpa de un confusor.
La causalidad sube una escalera de tres peldaños.	Asociación (ver), intervención (hacer) y contrafactuales (imaginar); cada uno exige más que el anterior.
El operador do borra las flechas que entran en la causa.	Intervenir es fijar X por la fuerza; un experimento aleatorizado lo consigue físicamente.
La fórmula de ajuste mide efectos a partir de datos.	Si el confusor está medido y cierra la puerta trasera, se promedian los efectos por subgrupo con los pesos de la población.
No todo lo medido se debe controlar.	Ajustar por un colisionador inventa correlación y ajustar por un mediador la borra; solo el grafo dice qué es un confusor.
La IA que solo correlaciona falla al intervenir.	Atajos, falta de robustez e injusticia nacen de confundir el primer peldaño con el segundo.

Para saber más

Koller, D. y Friedman, N. (2009). *Probabilistic Graphical Models: Principles and Techniques*. MIT Press.

Pearl, J. (2009). *Causality: Models, Reasoning, and Inference* (2.^a ed.). Cambridge University Press.

Pearl, J., Glymour, M. y Jewell, N. P. (2016). *Causal Inference in Statistics: A Primer*. Wiley.

Pearl, J. y Mackenzie, D. (2018). *The Book of Why: The New Science of Cause and Effect*. Basic Books.

Rubin, D. B. (1974). Estimating Causal Effects of Treatments in Randomized and Nonrandomized Studies. *Journal of Educational Psychology*, 66(5), 688-701.
<https://doi.org/10.1037/h0037350>

Russell, S. y Norvig, P. (2021). *Artificial Intelligence: A Modern Approach* (4.^a ed.). Pearson.

Spirtes, P., Glymour, C. y Scheines, R. (2000). *Causation, Prediction, and Search* (2.^a ed.). MIT Press.

Notas

1. Pearl, J. (2009). *Causality: Models, Reasoning, and Inference* (2.^a ed.). Cambridge University Press. Pearl formaliza la diferencia entre observar e intervenir mediante el operador `do`, demuestra el criterio de la puerta trasera y construye toda una jerarquía de preguntas causales que la teoría de la probabilidad clásica no sabía distinguir. ↵
2. Pearl, J., Glymour, M. y Jewell, N. P. (2016). *Causal Inference in Statistics: A Primer*. Wiley. El manual dedica un tratamiento detallado a la paradoja de Simpson y muestra que la pregunta «¿qué cifra es la correcta, la agregada o la de los subgrupos?» no se resuelve con más estadística, sino con un modelo causal que diga cuál es el confusor. ↵
3. Pearl, J. y Mackenzie, D. (2018). *The Book of Why: The New Science of Cause and Effect*. Basic Books. El libro divulga la escalera de la causalidad (asociación, intervención, contrafactuales) como metáfora central y argumenta que el aprendizaje automático basado solo en correlaciones se queda atrapado en el primer peldaño. ↵
4. Spirtes, P., Glymour, C. y Scheines, R. (2000). *Causation, Prediction, and Search* (2.^a ed.). MIT Press. Los autores desarrollan la teoría de los grafos causales y los algoritmos que intentan recuperar la estructura causal a partir de datos observacionales, estableciendo qué se puede y qué no se puede aprender sin intervenir. ↵
5. Rubin, D. B. (1974). Estimating Causal Effects of Treatments in Randomized and Nonrandomized Studies. *Journal of Educational Psychology*, 66(5), 688-701.
<https://doi.org/10.1037/h0037350>. Rubin escribe, para cada unidad, el resultado bajo tratamiento y bajo control, de los que solo uno se observa; el otro es el contrafactual, y de ahí surge el problema fundamental de la inferencia causal. ↵
6. Koller, D. y Friedman, N. (2009). *Probabilistic Graphical Models: Principles and Techniques*. MIT Press. El tratado conecta las redes bayesianas con los modelos causales y muestra que la intervención y los contrafactuales exigen información estructural que la distribución conjunta,

por sí sola, no determina. ↩

7. Russell, S. y Norvig, P. (2021). *Artificial Intelligence: A Modern Approach* (4.^a ed.). Pearson. El capítulo sobre redes causales presenta el operador `do`, la fórmula de ajuste y los modelos causales estructurales como la extensión natural de las redes bayesianas para razonar sobre acciones, y advierte de los riesgos de tratar un modelo predictivo como si entendiese causas.

↩

Capítulo 06: Razonamiento no monótono y revisión de creencias

Entrando en el tema

Te presento a Tweety. Te digo una sola cosa de él: es un pájaro. Sin pensarlo, ya has concluido algo que yo no he dicho: que vuela. No has consultado a Tweety, no has visto sus alas, no sabes su especie. Has aplicado una regla que llevas grabada desde niño, «los pájaros vuelan», y la has dado por buena. Es una conclusión razonable, útil y, casi siempre, correcta.

Ahora te doy un dato más: Tweety es un pingüino. En una décima de segundo retiras la conclusión anterior. Ya no crees que vuele; de hecho, ahora crees lo contrario. No has cambiado de regla, no has descubierto que los pájaros no vuelan. Simplemente has reconocido una excepción y has actualizado lo que creías. Y lo más interesante es lo que ha pasado con tu razonamiento: **una premisa nueva te ha hecho perder una conclusión que ya tenías**.

Eso, que para ti es trivial, es imposible en la lógica del capítulo 1. Allí la inferencia era **monótona**: si una conclusión se seguía de unas premisas, seguía siguiéndose por más premisas que añadieras. Añadir información solo podía hacer crecer el conjunto de cosas demostrables, nunca encogerlo. Es una propiedad maravillosa para garantizar verdades matemáticas, y un desastre para razonar sobre el mundo real, donde casi todo lo que sabemos es provisional, vale «por defecto» y está a una excepción de quedar desmentido.

Este capítulo trata de la lógica que **sí se retracta**. De cómo se formaliza «las aves vuelan, salvo las que no», cómo una base de conocimiento incorpora un hecho que contradice lo que creía sin volverse loca, y cómo razonar con argumentos que se atacan unos a otros. Es el puente entre la rigidez del capítulo 1 y el mundo lleno de excepciones de los sistemas expertos del capítulo 4, donde el pingüino que no vuela ya asomó como problema. Aquí lo formalizamos. Y al final veremos por qué esto reaparece, sin avisar, en cómo un modelo de lenguaje moderno maneja (a veces fatal) los casos por defecto y sus excepciones.

Monotonía frente a no monotonía: la lógica que no se retracta

Pongamos nombre exacto a la propiedad que rompe Tweety. En la lógica clásica, la relación de consecuencia cumple lo que se llama **monotonía**: ampliar las premisas nunca invalida una conclusión.

$$\text{Si } \Gamma \vdash \varphi \text{ entonces } \Gamma \cup \Delta \vdash \varphi$$

SÍMBOLO	SIGNIFICADO	EJEMPLO
Γ	El conjunto de premisas de partida.	{toda ave vuela, Tweety es ave}.
$\vdash$	«deriva»: a la izquierda hay con qué demostrar lo de la derecha.	Lo que separa lo que sabes de lo que se sigue.
φ	La conclusión derivada.	Tweety vuela.
Δ	Premisas nuevas que añadimos después.	{Tweety pesa 3 kg}.
$\cup$	Unión: juntar las premisas viejas con las nuevas.	Γ más Δ .

En palabras: si algo se deduce de un conjunto de premisas, se sigue deduciendo por muchas premisas más que le echés encima; la información nueva solo puede añadir conclusiones, jamás quitarlas.

Un ejemplo numérico de que esto es deseable en matemáticas: de $\{a > 5\}$ se deduce $a > 0$; si añadido $\{a \text{ es par}\}$, sigue siendo cierto que $a > 0$. Ninguna premisa adicional sobre a va a destruir esa desigualdad. Esa estabilidad es justo lo que quieres en una demostración: un teorema demostrado no deja de ser teorema porque aparezcan nuevos axiomas compatibles.

El sentido común funciona al revés.¹ Cuando concluyes «Tweety vuela» a partir de «es un ave», no estás afirmando una consecuencia lógica necesaria: estás haciendo una **conjetura razonable** que mantienes mientras nadie te lleve la contraria. Para marcar esa diferencia se usa otra flecha, la de la consecuencia **no monótona**, que se suele dibujar ondulada:

$$\Gamma \vdash \varphi \quad \text{pero} \quad \Gamma \cup \Delta \nvdash \varphi$$

En palabras: de lo que sé concluyo φ de forma plausible, pero al añadir Δ esa conclusión puede caerse; la flecha ondulada $\vdash$ avisa de que estamos razonando «por defecto», no con garantía de hierro.

Con $\Gamma = \{\text{las aves vuelan por defecto, Tweety es ave}\}$ concluimos $\varphi = \text{Tweety vuela}$. Pero con $\Delta = \{\text{Tweety es pingüino}\}$, la conclusión se retira. La flecha gruesa $\vdash$ nunca haría esto; la ondulada $\vdash$, sí. Todo el capítulo consiste en darle a esa flecha ondulada una definición precisa, porque «retractarse cuando toca» suena fácil hasta que intentas escribir las reglas exactas de cuándo toca.

Mundo cerrado y negación por fallo: lo que callas, lo niego

La forma más sencilla de razonamiento no monótono no nace de la filosofía, sino de la práctica: las bases de datos. Cuando consultas los vuelos de una compañía y el de las 14:30 a Lisboa no aparece, concluyes que **no existe**. Nadie ha escrito en ningún sitio «no hay vuelo a Lisboa a las 14:30»; lo deduces de su ausencia. A esa convención se le llama **suposición de mundo cerrado** (en inglés *closed-world assumption*, CWA): lo que no consta como verdadero se asume falso.

Es lo contrario de la lógica clásica, que vive en **mundo abierto**: si no puedo demostrar algo, no concluyo ni que es verdad ni que es mentira, simplemente no lo sé. El mundo cerrado es más atrevido y, para datos completos, mucho más útil: tu listado de vuelos es exhaustivo por diseño, así que la ausencia sí es información.

Esta idea tiene un nombre operativo en programación lógica: **negación por fallo**. Prolog no tiene una negación lógica de verdad; tiene un operador, `\+`, que significa «no consigo demostrarlo». La regla es: para evaluar `\+ Objetivo`, intenta probar `Objetivo`; si la prueba **falla**, entonces `\+ Objetivo` se da por cierto.

```
vuelo(madrid, paris, 0930).
vuelo(madrid, roma, 1145).

% "no hay vuelo a Lisboa" se concluye porque no se puede probar que lo haya
?- \+ vuelo(madrid, lisboa, _).
% Prolog responde: true
```

Aquí está la no monotonía, desnuda. Hoy `\+ vuelo(madrid, lisboa, _)` es verdadero. Si mañana alguien añade el hecho `vuelo(madrid, lisboa, 1430).`, la misma consulta pasa a ser falsa. Una premisa nueva ha retirado una conclusión: exactamente lo que la flecha gruesa del capítulo 1 tenía prohibido hacer. La negación por fallo es cómoda y rapidísima, pero arrastra un peligro que conviene tener grabado: **confunde «es falso» con «no lo sé»**. Si tu base de datos de vuelos está incompleta, concluirás que no existen vuelos que sí existen. El mundo cerrado solo es honesto cuando de verdad lo sabes todo sobre el dominio que cierras.

Lógica por defecto de Reiter: reglas con letra pequeña

La negación por fallo resuelve un caso, pero el razonamiento por defecto es más amplio: queremos poder decir «normalmente pasa esto, salvo que conste lo contrario» de forma general. En 1980, Raymond Reiter le dio a esa idea una forma matemática que sigue siendo la referencia: la **lógica por defecto**.²

Su pieza central es la **regla por defecto**, que se escribe como una fracción con tres partes:

$$\frac{\alpha : \beta}{\gamma}$$

SÍMBOLO	SIGNIFICADO	EJEMPLO
α	El prerrequisito : lo que debe constar para plantearse la regla.	$Ave(x)$.
β	La justificación : lo que debe ser consistente suponer.	$Vuela(x)$.
:	«y es consistente con todo lo demás suponer que...».	El filtro de excepciones.
γ	El consecuente : lo que se concluye si todo encaja.	$Vuela(x)$.

En palabras: si se cumple α y no hay nada en lo que ya sabes que contradiga β , entonces concluye γ ; en cuanto algo choque con β , la regla se calla y no concluye nada.

El caso de las aves es el ejemplo canónico, y se escribe así:

$$\frac{Ave(x) : Vuela(x)}{Vuela(x)}$$

Léelo despacio: «si x es un ave, y es consistente suponer que vuela, concluye que vuela». La justificación y el consecuente coinciden (esto se llama un *default normal*), y es la forma más común: la propia conclusión sirve de filtro. Mientras nada diga que x no vuela, la regla dispara. En cuanto sepamos que x **no** vuela, suponer que vuela deja de ser consistente, la regla no se aplica, y la conclusión «vuela» nunca llega a entrar.

Añadamos el conocimiento del pingüino como hechos clásicos, no como defaults, porque no son conjeturas sino verdades duras:

- $Pingüino(x) \rightarrow Ave(x)$: todo pingüino es un ave.
- $Pingüino(x) \rightarrow \neg Vuela(x)$: ningún pingüino vuela.

Ahora razonemos sobre dos individuos. Sobre **Tweety, del que solo sabemos que es ave**: el prerrequisito $Ave(tweety)$ se cumple, y suponer $Vuela(tweety)$ es consistente con todo lo que sabemos, así que la regla dispara y concluimos que vuela. Sobre **Pingu, del que sabemos que es pingüino**: de $Pingüino(pingu)$ se sigue $Ave(pingu)$, así que el prerrequisito también se cumple; pero también se sigue $\neg Vuela(pingu)$, con lo que suponer $Vuela(pingu)$ es **inconsistente**. La justificación falla, la regla se bloquea, y la conclusión sobre Pingu es que no vuela. La misma regla, dos resultados, sin contradicción.

Una premisa nueva retira una conclusión

Lo que la lógica monótona del capítulo 1 tiene prohibido hacer, el sentido común lo hace a diario.

ESTADO INICIAL

Sé: Tweety es un ave
regla por defecto: las aves vuelan

Concluyo: Tweety vuela
provisional, mientras nada lo niegue

LLEGA INFORMACIÓN
Tweety es un pingüino
y los pingüinos no vuelan

ESTADO REVISADO

~~Tweety vuela~~
conclusión retirada

Concluyo: Tweety no vuela
la excepción manda

La regla «las aves vuelan» no ha cambiado: ha cambiado lo que sabemos de Tweety.
Eso es razonamiento no monótono: conjeturar de más y corregir cuando hace falta.

IA para gente curiosa / Facsímil 12 / Capítulo 06 / 686f6c61

Extensiones: las visiones coherentes del mundo

Una teoría por defecto puede producir más de un mundo coherente. A cada uno de esos mundos cerrados y consistentes que se obtienen aplicando todos los defaults que se pueden aplicar, Reiter lo llama una **extensión**: un conjunto de conclusiones que está cerrado bajo la lógica clásica y que respeta las justificaciones de los defaults usados. En el caso de las aves, la extensión sobre Tweety contiene «vuela»; la extensión sobre Pingu contiene «no vuela».

El detalle que vuelve esto interesante, y a veces incómodo, es que **puede haber varias extensiones incompatibles**. El ejemplo clásico es Nixon, que era a la vez cuáquero (y los cuáqueros son pacifistas por defecto) y republicano (y los republicanos no lo son por defecto). Las dos reglas dan resultados opuestos, ninguna excepción explícita decide, y la teoría tiene dos extensiones: una en la que Nixon es pacifista y otra en la que no. La lógica por defecto no inventa una respuesta única; te muestra que el conocimiento, tal como está escrito, admite dos posturas razonables. Ese «el sistema puede tener varios modelos coherentes a la vez» es exactamente la idea que más adelante reaparece, con maquinaria moderna, en Answer Set Programming.

Circunscripción de McCarthy: minimizar lo anormal

El mismo año que Reiter, John McCarthy atacó el problema desde otro ángulo, más semántico que sintáctico: la **circunscripción**.³ Su intuición es preciosa y muy humana: **suponer que el mundo es lo más normal posible**. Las cosas son corrientes salvo que conste que son raras.

La maquinaria consiste en introducir un predicado explícito de anormalidad, *Ab* (de *abnormal*), y escribir las reglas con una cláusula de escape. «Las aves vuelan» se convierte en «las aves vuelan salvo que sean anormales en este aspecto»:

$$\forall x (Ave(x) \wedge \neg Ab(x) \rightarrow Vuela(x))$$

En palabras: sea quien sea ese x , si es un ave y no es anormal, entonces vuela; la coletilla $\neg Ab(x)$ es la puerta por la que se cuelan las excepciones.

La parte no monótona es la receta para decidir quién es anormal: **circunscribir** Ab significa quedarse con los modelos en los que el conjunto de cosas anormales es lo más pequeño posible. Si nada te obliga a declarar anormal a Tweety, no lo declaras, y vuela. Pero si añades «los pingüinos son anormales respecto a volar» ($Pingüino(x) \rightarrow Ab(x)$) y sabes que Pingu es pingüino, entonces Pingu **tiene** que ser anormal, no hay forma de minimizarlo fuera, y la regla del vuelo no se le aplica. La no monotonía está en que minimizar es una operación que el conocimiento nuevo puede deshacer: una premisa que fuerza una anormalidad nueva cambia cuál es el conjunto mínimo y, con él, las conclusiones.

Reiter y McCarthy llegan al mismo sitio por caminos distintos. Reiter escribe reglas con letra pequeña y construye extensiones; McCarthy escribe reglas con una válvula de anormalidad y minimiza esa válvula. Los dos capturan la misma idea de fondo: **lo normal por defecto, lo raro solo cuando consta.**

Revisión de creencias: cambiar de opinión con método

Hasta aquí hemos visto cómo concluir cosas provisionales. Falta la otra mitad: qué hacer cuando llega un hecho que **contradice de frente** lo que ya creías. Tu base de conocimiento decía «todos los cisnes son blancos» y aparece un cisne negro. No puedes simplemente añadir el hecho nuevo, porque entonces tu base afirmaría a la vez que ese cisne es blanco y que es negro, y de una contradicción la lógica clásica deduce cualquier cosa. Tienes que **reacomodar** lo que sabes. La teoría que estudia cómo hacerlo con orden se conoce por las iniciales de sus tres autores: **AGM**, por Alchourrón, Gärdenfors y Makinson.⁴

AGM distingue tres operaciones sobre una base de creencias K , y la diferencia entre ellas es la frase clave de toda la sección:

OPERACIÓN	SÍMBOLO	QUÉ HACE	CUÁNDO SE USA
Expansión	$K + \varphi$	Añade φ sin tocar nada más.	Cuando φ no contradice lo que ya sabes.
Revisión	$K * \varphi$	Añade φ y repara lo que haga falta para no caer en contradicción.	Cuando φ choca con lo que crees.
Contracción	$K \div \varphi$	Quita φ (y lo justo para que no se vuelva a deducir).	Cuando dejas de estar seguro de φ .

La expansión es el caso fácil y monótono: si te enteras de algo compatible, lo apuntas y ya está. El caso difícil es la revisión, y AGM la resuelve con una idea elegante, la **identidad de Levi**: revisar es contraer primero y expandir después.

$$K * \varphi = (K \div \neg\varphi) + \varphi$$

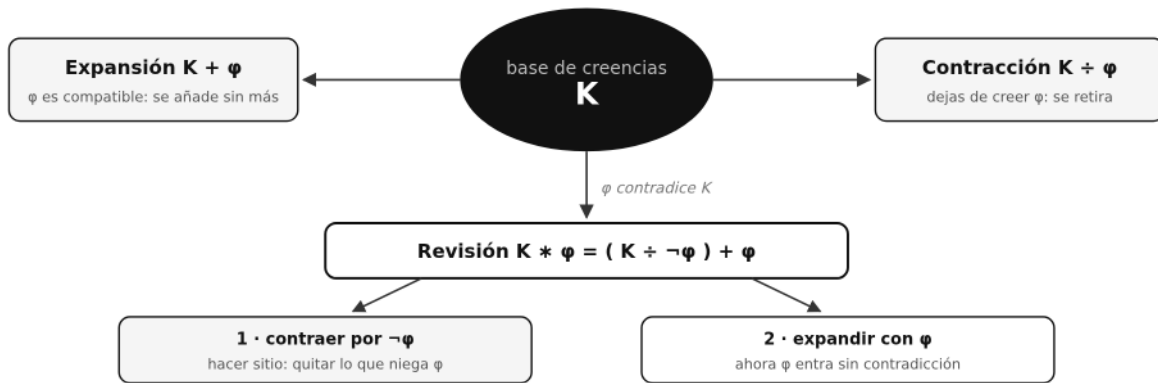
SÍMBOL O	SIGNIFICADO	EJEMPLO
K	La base de creencias actual.	«Todos los cisnes son blancos».
$*$	Revisión: incorporar φ manteniendo la coherencia.	Meter «este cisne es negro» sin romperlo todo.
$\div$	Contracción: retirar una creencia.	Soltar «todos los cisnes son blancos».
$+$	Expansión: añadir sin más.	Apuntar el cisne negro una vez hay sitio.
$\neg\varphi$	La negación de lo que queremos incorporar.	«No existe ningún cisne no blanco».

En palabras: para incorporar un hecho que te contradice, primero quita de tu base todo lo que se opone a él (contrae por su negación) y solo entonces añádelo; nunca metas la novedad encima de la creencia que la niega, porque eso te deja con las dos a la vez.

Veámoslo con el cisne. Tu base K contiene «todos los cisnes son blancos», de donde se deduce «este cisne es blanco». Llega φ = «este cisne es negro», así que $\neg\varphi$ = «este cisne es blanco». La identidad de Levi dice: primero $K \div \neg\varphi$, es decir, retira la creencia «este cisne es blanco» (y para que no se vuelva a deducir, tienes que debilitar también «todos los cisnes son blancos», quizá a «casi todos los cisnes son blancos»). Una vez tu base ya no afirma que el cisne sea blanco, expandes con «este cisne es negro». Resultado: una base coherente que reconoce la excepción sin haber explotado por el camino.

Las tres formas de cambiar de opinión

Una base de creencias se actualiza añadiendo, quitando o reparando, según el hecho que llegue.



IA para gente curiosa / Facsimil 12 / Capítulo 06 / 686f6c61

Los postulados: qué cuenta como un cambio racional

Lo que hace memorable a AGM no es la receta, sino que fijaron por escrito **qué propiedades debe cumplir cualquier forma sensata de revisar**, sin decir cómo implementarla. Son los **postulados de racionalidad**, y en lenguaje llano dicen cosas que, una vez enunciadas, parecen de cajón:

- **Éxito.** Después de revisar por φ , tu base cree φ . Si te has tomado la molestia de incorporar un hecho, que conste que lo crees.
- **Coherencia.** Si φ no es una contradicción en sí misma, la base revisada es consistente. Cambiar de opinión no debe dejarte creyendo cualquier cosa.
- **Inclusión.** Revisar no añade nada que no estuviera ya implícito al expandir; no te inventas creencias de la nada.
- **Cambio mínimo.** Y el más importante de todos: al reparar, **tira lo menos posible**. Si para aceptar el cisne negro basta con soltar «todos los cisnes son blancos», no aproveches para olvidar también dónde aparcaste el coche. La economía cognitiva es la regla de oro: conserva todo lo que puedas conservar.

Ese principio de cambio mínimo es el que conecta AGM con todo lo anterior. Tanto los defaults de Reiter como la circunscripción de McCarthy son, en el fondo, formas de «cambiar lo menos posible» ante información nueva: mantén tus conjeturas normales mientras puedas, y retoca solo lo que la realidad te obligue a retocar.

Argumentación: cuando las razones se pelean

Hay una última manera de domar la no monotonía, y es quizá la más cercana a cómo discutimos los humanos: no calcular extensiones ni contraer bases, sino **poner los argumentos a competir** y ver cuáles sobreviven. En 1995, Phan Minh Dung dio a esta idea

un marco abstracto de una sencillez sorprendente.⁵

Un **marco de argumentación** es asombrosamente austero: un conjunto de argumentos y una relación de ataque entre ellos. Nada más. No importa qué dice cada argumento por dentro; solo importa quién ataca a quién.

$$AF = \langle \mathcal{A}, \mathcal{R} \rangle, \quad \mathcal{R} \subseteq \mathcal{A} \times \mathcal{A}$$

SÍMBOL O	SIGNIFICADO	EJEMPLO
$\mathcal{A}$	El conjunto de argumentos en juego.	«Tweety vuela porque es ave».
$\mathcal{R}$	La relación de ataque: qué argumento contradice a cuál.	«Tweety es pingüino» ataca al anterior.
$\langle \rangle$	El marco completo: argumentos más sus ataques.	El debate entero, dibujado como un grafo.

En palabras: un marco de argumentación es un mapa de un debate donde cada flecha significa «este argumento tumba a aquel»; razonar es decidir qué grupo de argumentos puede sostenerse a la vez sin atacarse y defendiéndose de todo lo que les ataca.

La clave es la noción de **defensa**: un argumento está defendido por un conjunto si, para cada ataque que recibe, alguien del conjunto contraataca al atacante. A partir de ahí, Dung define conjuntos de argumentos que se aceptan juntos, las **extensiones** (el nombre vuelve, no por casualidad). Una extensión debe ser **libre de conflicto** (sus miembros no se atacan entre sí) y **admisibile** (defiende a todos sus miembros). Sobre Tweety, el argumento «es pingüino, no vuela» ataca a «es ave, vuela» y no recibe contraataque, así que sobrevive solo: la extensión aceptada concluye que no vuela. La excepción gana el debate.

Lo verdaderamente potente del resultado de Dung es que este marco minimalista **engloba a casi todos los anteriores**: la lógica por defecto, la negación por fallo de la programación lógica y otras formas de razonamiento no monótono se pueden traducir a marcos de argumentación. Detrás de «calcula extensiones» y «contrae tu base» late, una y otra vez, la misma pregunta: ¿qué conjunto de afirmaciones puede sostenerse coherentemente frente a las objeciones?

Las herramientas que lo ejecutan

Nada de esto es solo pizarra. Hay motores reales que razonan así todos los días:

- **Prolog**, con su negación por fallo (`\+`), es la puerta de entrada práctica al mundo cerrado y la base de incontables sistemas de reglas.

- **Answer Set Programming** (ASP) es la heredera moderna y más expresiva. Se apoya en la **semántica de modelos estables** de Gelfond y Lifschitz, que da sentido preciso a la negación no monótona y permite que un programa tenga varios modelos coherentes, igual que una teoría por defecto tiene varias extensiones.⁶ Solvers como **clingo** resuelven con ella problemas de planificación, configuración y diagnóstico donde las excepciones son la norma.
- Para la revisión de creencias hay razonadores específicos que implementan operadores AGM, y la idea vive además dentro de cualquier sistema que tenga que mantener una base de conocimiento coherente mientras llegan hechos que se contradicen.

Para un mapa de conjunto de toda esta familia, el texto de referencia sigue siendo el panorama de Brewka, Dix y Konolige, que ordena defaults, circunscripción, argumentación y semántica de modelos estables bajo un mismo techo.⁷

En el día a día

Aunque no escribas nunca una regla por defecto ni la identidad de Levi, esta maquinaria está actuando bajo cosas que tocas a diario.

Cada vez que tu móvil asume que sigues en tu ciudad porque no le consta que hayas viajado, está haciendo mundo cerrado. Cada vez que un sistema de permisos concede acceso «por defecto al equipo, salvo excepción explícita para este usuario», está aplicando una regla con su letra pequeña, exactamente la estructura del default de Reiter. Cada vez que un GPS recalcula la ruta porque te has saltado una salida, está revisando sus creencias: tenía un plan, llegó un hecho que lo contradecía, y reparó lo justo sin tirar el destino. Y cada vez que un corrector o un asistente «cambia de idea» a mitad de frase porque una palabra posterior aclara el sentido, está haciendo algo que se parece, de lejos, a retractar una conclusión provisional.

En los sistemas expertos del capítulo 4, la no monotonía es la sangre de las excepciones: la regla de especificidad que hacía ganar a «si es ave y pingüino, no vuela» sobre «si es ave, vuela» es razonamiento por defecto disfrazado de prioridad. Y en las bases de conocimiento que evolucionan (un grafo de empresa, una ontología médica que incorpora un fármaco nuevo que contraindica a otro), la revisión de creencias es lo que evita que la base se vuelva incoherente cada vez que el mundo cambia.

Por qué debería importarte

Porque la diferencia entre un sistema que parece inteligente y uno que de verdad lo aguanta está, muchas veces, en cómo maneja las excepciones y la información que falta. Un sistema que solo sabe lógica monótona es frágil: en cuanto el mundo le presenta un pingüino, o se

queda mudo o se contradice. Un sistema que razona por defecto sigue siendo útil con información incompleta y se corrige con elegancia cuando llega el dato que faltaba.

Esto conecta de lleno con los modelos de lenguaje. Un LLM está empapado de defaults estadísticos: «un ave vuela», «un médico es un hombre», «un cumpleaños se celebra». Esos defaults lo hacen sonar razonable, y son justo los que arrastra mal cuando aparece la excepción: tiende a no retractarse con limpieza, a mantener la conjetura por defecto aunque el contexto la haya desmentido tres líneas antes, o a no distinguir «es falso» de «no me consta». Entender la diferencia entre una conclusión monótona (garantizada) y una por defecto (revisable) te da el vocabulario para saber cuándo exigirle a un sistema una garantía dura y cuándo aceptar una conjetura blanda, y para diseñar la capa que se encarga de retractarse cuando el modelo no sabe hacerlo solo.

Dónde solía tropezar yo

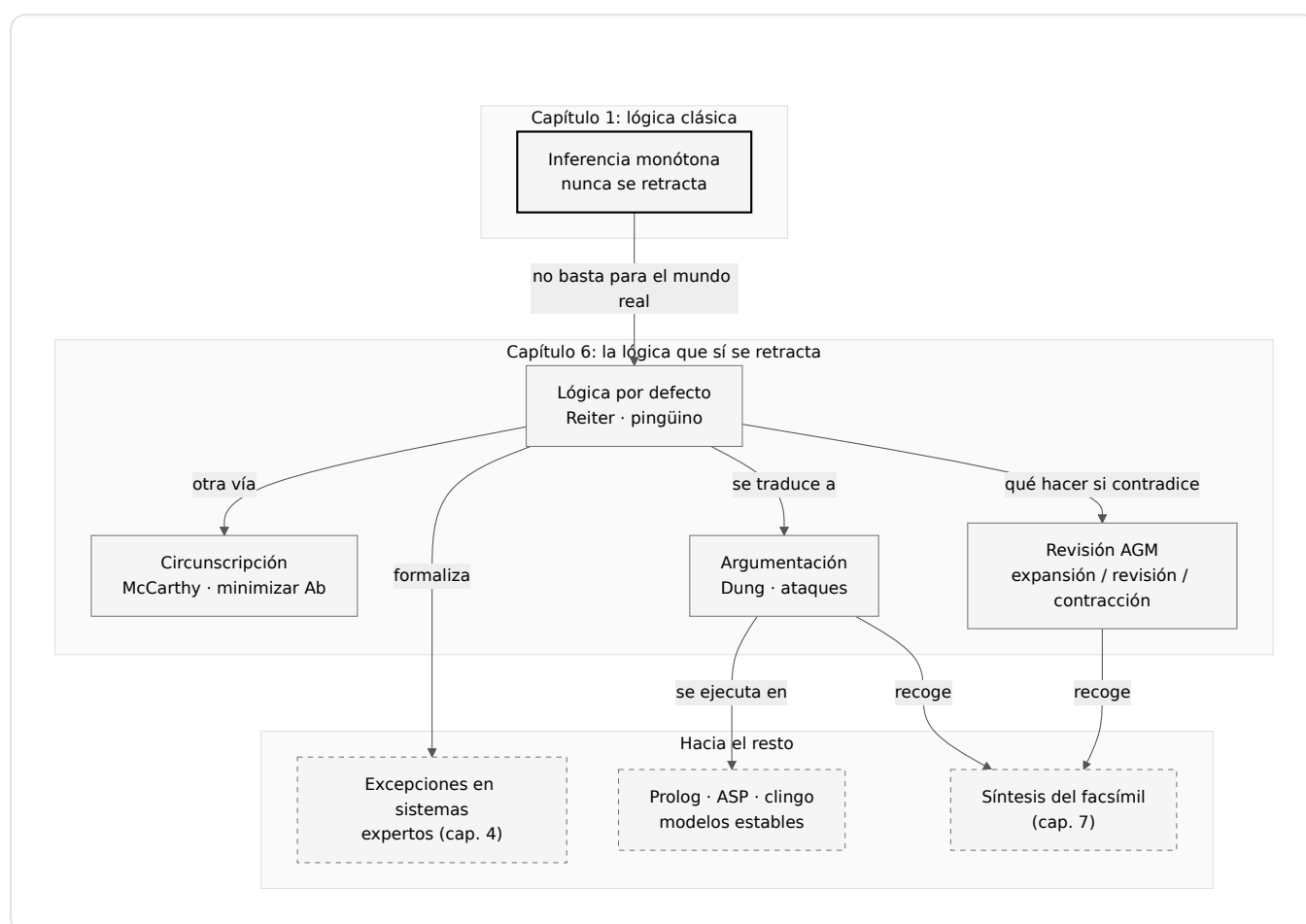
ERROR	POR QUÉ ES UN ERROR	ANTÍDOTO
Confundir «es falso» con «no lo puedo probar»	La negación por fallo concluye falso desde la ausencia. Si la base está incompleta, niegas cosas que sí son verdad.	Antes de cerrar el mundo, pregúntate si de verdad lo sabes todo sobre ese dominio. Si no, usa mundo abierto.
Esperar una única extensión	Una teoría por defecto puede tener varias visiones coherentes e incompatibles (el caso Nixon).	Acepta que a veces el conocimiento, tal como está escrito, admite varias posturas; decide qué política sigues (crédula o escéptica).
Revisar añadiendo encima de la contradicción	Si metes φ sin quitar $\neg\varphi$, tu base afirma las dos y se vuelve inconsistente.	Aplica la identidad de Levi: contrae por $\neg\varphi$ primero, expande con φ después.
Tirar demasiado al revisar	Para aceptar un hecho nuevo, olvidas medio sistema «por si acaso».	Respetar el cambio mínimo: retira solo lo que de verdad choca con la novedad.
Creer que un default es una verdad	«Las aves vuelan» es una conjetura revisable, no un teorema. Tratarla como ley te impide ver la excepción.	Marca mentalmente qué conclusiones son por defecto (flecha ondulada) y cuáles son lógicas duras (flecha gruesa).

Cómo encaja todo

Este capítulo es la bisagra del facsímil. El capítulo 1 te dio una lógica perfecta y rígida: correcta, completa y monótona, incapaz de retractarse de nada. Era exactamente lo que necesitabas para demostrar teoremas, y exactamente lo que te falla en cuanto sales a la calle,

donde casi todo lo que sabes vale «por defecto». Lo que aquí hemos hecho es añadirle a esa lógica la capacidad de cambiar de opinión: reglas con letra pequeña (Reiter), una válvula de anomalía que se minimiza (McCarthy), un método para reparar la base cuando un hecho la contradice (AGM) y un modo de poner los argumentos a competir (Dung).

El hilo más corto va al **capítulo 4**, el de los sistemas expertos. Allí, la regla de especificidad hacía que «si es ave y pingüino, no vuela» ganara a «si es ave, vuela», y se quedó como un truco del motor. Aquí ese truco tiene nombre y teoría: es razonamiento por defecto. Las excepciones que volvían frágiles a XCON y a MYCIN son, vistas desde este capítulo, defaults que se bloquean y bases de conocimiento que hay que revisar sin romper. Y el hilo hacia delante lleva al **capítulo 7**, donde recogeremos todo el facsímil: la lógica dura del capítulo 1, la probabilidad de los capítulos intermedios y esta lógica blanda que se retracta, tres formas complementarias de que una máquina razone con honestidad sobre un mundo incompleto.



Vocabulario aprendido

TÉRMINO	DEFINICIÓN
Razonamiento monótono	Razonamiento en el que añadir información nunca retira una conclusión previa.
Razonamiento no monótono	Razonamiento que puede retractar conclusiones al llegar información nueva.
Suposición de mundo cerrado	Asumir falso todo lo que no se puede demostrar verdadero.
Negación por fallo	Concluir la negación de algo porque no se logra demostrarlo, como en Prolog.
Lógica por defecto	Lógica de Reiter con reglas que valen salvo excepción, mientras sea consistente suponerlas.
Extensión	Conjunto de conclusiones coherente que se obtiene aplicando los defaults posibles.
Circunscripción	Método de McCarthy que minimiza lo anormal para razonar por defecto.
Revisión de creencias	Incorporar a una base de conocimiento un hecho nuevo manteniéndola consistente.
AGM	Marco de Alchourrón, Gärdenfors y Makinson para expansión, revisión y contracción de creencias.
Argumentación	Razonar con argumentos que se atacan entre sí y decidir cuáles son aceptables.
Answer Set Programming	Programación lógica no monótona basada en modelos estables para resolver problemas con reglas y excepciones.

Antes de pasar página

- ☐ ¿Sé enunciar la monotonía y explicar con Tweety por qué el sentido común la rompe? (Si no, vuelve a «Monotonía frente a no monotonía».)
- ☐ ¿Distingo «es falso» de «no lo puedo probar» y veo el riesgo de la negación por fallo? (Si no, vuelve a «Mundo cerrado y negación por fallo».)
- ☐ ¿Puedo leer una regla por defecto $\frac{\alpha:\beta}{\gamma}$ y decir cuándo se bloquea ante una excepción? (Si no, vuelve a «Lógica por defecto de Reiter».)
- ☐ ¿Entiendo qué es una extensión y por qué puede haber varias incompatibles? (Si no, vuelve a «Extensiones».)

- ☐ ¿Sé qué minimiza la circunscripción y cómo se cuela una excepción por $\neg Ab$? (Si no, vuelve a «Circunscripción de McCarthy».)
- ☐ ¿Distingo expansión, revisión y contracción, y sé aplicar la identidad de Levi? (Si no, vuelve a «Revisión de creencias».)
- ☐ ¿Entiendo el principio de cambio mínimo y por qué es el postulado clave? (Si no, vuelve a «Los postulados».)
- ☐ ¿Puedo dibujar un marco de argumentación y decir qué argumento sobrevive? (Si no, vuelve a «Argumentación».)

En resumen

IDEA FUERZA	DETALLE
La lógica clásica es monótona; el sentido común no.	Añadir premisas nunca retira conclusiones en el capítulo 1; en el mundo real, un pingüino te hace retractarte.
Lo que callas, el mundo cerrado lo niega.	La negación por fallo es rápida y útil, pero confunde «falso» con «no me consta»; solo es honesta si lo sabes todo.
Los defaults son reglas con letra pequeña.	$\frac{\alpha:\beta}{\gamma}$ concluye γ salvo que la excepción haga inconsistente la justificación; pueden surgir varias extensiones.
Revisar es contraer y luego expandir.	La identidad de Levi incorpora un hecho que contradice la base quitando primero su negación y cambiando lo mínimo.
Esta lógica blanda sostiene la IA robusta.	Excepciones en sistemas expertos, bases que evolucionan y los defaults que los LLM manejan mal viven todos aquí.

Para saber más

Alchourrón, C. E., Gärdenfors, P. y Makinson, D. (1985). On the logic of theory change: Partial meet contraction and revision functions. *Journal of Symbolic Logic*, 50(2), 510-530. <https://doi.org/10.2307/2274239>

Brewka, G., Dix, J. y Konolige, K. (1997). *Nonmonotonic Reasoning: An Overview*. CSLI Publications.

Dung, P. M. (1995). On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial Intelligence*, 77(2), 321-357. [https://doi.org/10.1016/0004-3702\(94\)00041-X](https://doi.org/10.1016/0004-3702(94)00041-X)

Gelfond, M. y Lifschitz, V. (1988). The stable model semantics for logic programming. En *Proceedings of the Fifth International Conference and Symposium on Logic Programming* (pp. 1070-1080). MIT Press.

McCarthy, J. (1980). Circumscription—a form of non-monotonic reasoning. *Artificial Intelligence*, 13(1-2), 27-39. [https://doi.org/10.1016/0004-3702\(80\)90011-9](https://doi.org/10.1016/0004-3702(80)90011-9)

Reiter, R. (1980). A logic for default reasoning. *Artificial Intelligence*, 13(1-2), 81-132. [https://doi.org/10.1016/0004-3702\(80\)90014-4](https://doi.org/10.1016/0004-3702(80)90014-4)

Russell, S. y Norvig, P. (2021). *Artificial intelligence: a modern approach* (4.^a ed.). Pearson.

Notas

1. Russell, S. y Norvig, P. (2021). *Artificial intelligence: a modern approach* (4.^a ed.). Pearson. Los capítulos de representación del conocimiento presentan el razonamiento no monótono, la lógica por defecto y la circunscripción como respuestas al carácter incompleto y revisable del conocimiento de sentido común. ↵
2. Reiter, R. (1980). A logic for default reasoning. *Artificial Intelligence*, 13(1-2), 81-132. [https://doi.org/10.1016/0004-3702\(80\)90014-4](https://doi.org/10.1016/0004-3702(80)90014-4) El artículo introduce las reglas por defecto y la noción de extensión, y es el texto fundacional del razonamiento no monótono formal. ↵
3. McCarthy, J. (1980). Circumscription—a form of non-monotonic reasoning. *Artificial Intelligence*, 13(1-2), 27-39. [https://doi.org/10.1016/0004-3702\(80\)90011-9](https://doi.org/10.1016/0004-3702(80)90011-9) McCarthy propone razonar por defecto minimizando la extensión de un predicado de anormalidad, formalizando la idea de que las cosas son normales salvo prueba en contra. ↵
4. Alchourrón, C. E., Gärdenfors, P. y Makinson, D. (1985). On the logic of theory change: Partial meet contraction and revision functions. *Journal of Symbolic Logic*, 50(2), 510-530. <https://doi.org/10.2307/2274239> El artículo fundacional de la revisión de creencias define las tres operaciones y los postulados de racionalidad que deben cumplir. ↵
5. Dung, P. M. (1995). On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial Intelligence*, 77(2), 321-357. [https://doi.org/10.1016/0004-3702\(94\)00041-X](https://doi.org/10.1016/0004-3702(94)00041-X) El artículo introduce los marcos de argumentación abstracta y demuestra que unifican buena parte del razonamiento no monótono previo. ↵
6. Gelfond, M. y Lifschitz, V. (1988). The stable model semantics for logic programming. En *Proceedings of the Fifth International Conference and Symposium on Logic Programming* (pp. 1070-1080). MIT Press. Define la semántica de modelos estables, fundamento de Answer Set Programming. ↵

7. Brewka, G., Dix, J. y Konolige, K. (1997). *Nonmonotonic Reasoning: An Overview*. CSLI Publications. Una panorámica unificada de los principales formalismos no monótonos y sus relaciones. ↵

Cuadernos para practicar

Escanea el código con el móvil para abrir el cuaderno en Google Colab y ejecutarlo sin instalar nada.



Razonamiento no monótono: retractarse cuando llega un dato nuevo

<https://colab.research.google.com/github/686f6c61/iaparagentecuriosa-notebooks/blob/main/Facs%C3%ADmil%2012/06-razonamiento-no-monotono.ipynb>

Capítulo 07: Lo que deberías saber: lógica, conocimiento e incertidumbre

Entrando en el tema

Este facsímil ha recorrido la otra mitad de la inteligencia artificial: no la que aprende de millones de ejemplos, sino la que **razona con reglas, probabilidades y grados**. Empezamos con la lógica, que decide con verdades tajantes; vimos que el mundo real rara vez es tan limpio y pasamos a la probabilidad y a Bayes, que miden la creencia; admitimos que muchos conceptos no son ni ciertos ni falsos, sino graduales, y entró la lógica difusa; juntamos las piezas en los sistemas expertos, que codifican el saber de un dominio en reglas que un motor de inferencia recorre y explica; y terminamos con el razonamiento causal, el salto de «qué va con qué» a «qué pasa si intervengo».

Este capítulo de cierre hace dos cosas: recapitula las ideas para que las tengas juntas, mostrando cómo las cinco herramientas trabajan sobre un mismo problema, y te entrega los cuadernos para ejecutar cada pieza con tus propias manos.

Recapitulación activa del facsímil

1. Lógica para máquinas: proposicional, primer orden y resolución

La lógica es la forma más antigua de hacer que una máquina deduzca. Vimos la **lógica proposicional** (enunciados verdaderos o falsos combinados con conectivas), las **tablas de verdad** para decidir validez y satisfacibilidad, y la **regla de resolución** con su prueba por refutación: para demostrar algo, niégalo y deriva una contradicción. Subimos a la **lógica de primer orden**, con predicados, cuantificadores y **unificación**, que es lo que permite a un mini-Prolog encadenar reglas. Y enlazamos con el SAT del facsímil 2: la misma satisfacibilidad booleana, vista desde la lógica.

2. Razonamiento bajo incertidumbre: Bayes y redes bayesianas

Cuando los hechos son inciertos, la verdad tajante no sirve. La **probabilidad** mide el grado de creencia y el **teorema de Bayes** dice cómo actualizarla al ver evidencia. Aprendimos por qué un test positivo en una enfermedad rara deja la sospecha sorprendentemente baja (la tasa base manda), montamos un **clasificador naive Bayes** y subimos a las **redes bayesianas**: un grafo que factoriza una distribución conjunta enorme en tablas pequeñas y permite preguntar cosas como «¿cuál es la probabilidad de robo si suena la alarma?».

3. Cuando la verdad es difusa: factores de certeza y lógica difusa

No toda incertidumbre es probabilística: hay conceptos **vagos** («calentito», «alto», «caro») que se capturan con grados de pertenencia, no con probabilidades. La **lógica difusa** de Zadeh y los **factores de certeza** de MYCIN modelan ese terreno. Recorrimos la inferencia de **Mamdani** de principio a fin —fuzzificar, evaluar reglas, agregar y **defuzzificar** por centroide — y vimos que un controlador difuso produce respuestas suaves donde un sistema de umbrales daría saltos bruscos.

4. Sistemas expertos y soporte a la decisión

Un **sistema experto** codifica el saber de un dominio en una base de reglas que un **motor de inferencia** recorre, sea **hacia delante** (de los datos a las conclusiones) o **hacia atrás** (de la hipótesis a los hechos que la sostienen). Vimos su arquitectura, la resolución de conflictos, la idea del algoritmo Rete, los sistemas clásicos (MYCIN, DENDRAL, XCON) y su fragilidad, y la diferencia con un **sistema de soporte a la decisión**, que asiste a una persona en vez de decidir por ella.

5. Razonamiento causal: de la correlación a la intervención

Dimos el salto que la probabilidad sola no da: de la correlación a la causa. Con la **paradoja de Simpson** vimos cómo una tendencia se invierte al estratificar por un confusor; con la **escalera de la causalidad** separamos ver, hacer e imaginar; y con el **operador do** y la **fórmula de ajuste por puerta trasera** aprendimos a estimar el efecto real de una intervención, no solo la asociación observada. Es la herramienta que distingue un modelo que predice de uno que entiende qué pasa al actuar.

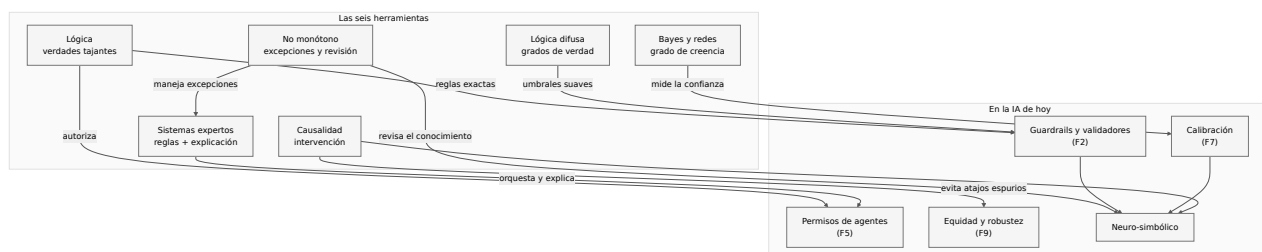
6. Razonamiento no monótono y revisión de creencias

Y cerramos con la lógica que **se retracta**. La lógica clásica del capítulo 1 es monótona: una vez que concluye algo, ninguna información nueva se lo quita. El sentido común no funciona así: concluyes que «Tweety vuela» porque es un ave, y te retractas al saber que es un pingüino. Vimos la **negación por fallo**, la **lógica por defecto** de Reiter (reglas que valen salvo excepción), la **revisión de creencias** AGM (cómo una base de conocimiento incorpora un hecho que contradice lo anterior sin volverse inconsistente) y la **argumentación**. Es lo que permite a un sistema razonar con información incompleta y corregirse, justo lo que un mundo cambiante exige.

Cómo encaja todo

Estas seis piezas no son una reliquia de museo: son la otra mitad que, sin que la veas, sostiene la inteligencia artificial que usas a diario. Cuanto más potente y más opaco es un modelo, más falta hacen las herramientas que saben decir, con claridad, «esto es seguro», «esto es probable», «esto es cuestión de grado» y «esto causa aquello». Un modelo de

lenguaje es, por dentro, una máquina **probabilística**; su **calibración** (facsimilar 7) es Bayes aplicado a sus salidas. Los **guardrails** y validadores (facsimilar 2) son **lógica**: reglas exactas que aceptan o rechazan. Los **permisos** de un agente (facsimilar 5) son un pequeño **sistema experto**. Y la promesa a medio plazo —sistemas que aprenden y explican— es el matrimonio **neuro-simbólico**: la red percibe y propone, la lógica y la causalidad comprueban y justifican.



Cuadernos para practicar

Has modelado un problema con las cinco herramientas del facsimilar; estos cuadernos te dejan ejecutar cada pieza. Son *notebooks* que se abren en Google Colab —gratis, en el navegador— o te puedes descargar. Cada uno está explicado paso a paso, con salidas reales que se generan al pulsar el botón de ejecutar, y dice de qué capítulo sale.

Lógica y resolución: deducir como una máquina

Qué prácticas: construir tablas de verdad, demostrar argumentos por resolución y unificar términos de primer orden. **Dónde encaja:** capítulo 1 (lógica proposicional, primer orden y resolución).

Montas las herramientas de la lógica desde cero. Evalúas una fórmula sobre sus mundos posibles, demuestras un argumento clásico **por refutación** derivando la cláusula vacía, y compruebas el coste real de cada enfoque: con 18 variables, la tabla de verdad tiene **262.144** filas que mirar, mientras que un mini-DPLL decide lo mismo con **8** decisiones. Para rematar, implementas la **unificación** y un mini-Prolog que responde una consulta encadenando reglas.

► [Abrir en Google Colab](#)

↓ [Descargar el cuaderno \(.ipynb\)](#)

Bayes y redes bayesianas: actualizar creencias con datos

Qué prácticas: aplicar el teorema de Bayes, entrenar un naive Bayes y consultar una red bayesiana. **Dónde encaja:** capítulo 2 (probabilidad, Bayes y redes bayesianas).

Empiezas con la trampa de la tasa base: un test positivo para una enfermedad rara deja la probabilidad de estar enfermo en apenas un **16,7 %**, y ves cómo cambia al variar la prevalencia. Luego entrenas un **naive Bayes** y montas la **red bayesiana de la alarma**: la probabilidad de robo cuando llaman los dos vecinos sube del **0,1 %** de partida al **28,4 %**; y compruebas el *explaining away*, cuando un terremoto «explica» la alarma y la sospecha de robo se desploma del **37,4 %** al **0,33 %**.

► [Abrir en Google Colab](#)

↓ [Descargar el cuaderno \(.ipynb\)](#)

Lógica difusa: cuando la verdad es cuestión de grado

Qué prácticas: definir conjuntos difusos y ejecutar un controlador de Mamdani de principio a fin. **Dónde encaja:** capítulo 3 (factores de certeza y lógica difusa).

Defines funciones de pertenencia y ves que **26 grados** pertenecen a la vez a «templado» con grado **0,607** y a «caliente» con grado **0,333**. Combinas evidencias con factores de certeza y montas un **controlador difuso** completo (la propina de un restaurante según servicio y comida): fuzzificas, evalúas las reglas, agregas y **defuzzificas** por centroide hasta un número concreto —una propina del **14,81 %**—. Y lo comparas con un controlador de umbrales para ver la diferencia entre una respuesta suave y una a saltos.

► [Abrir en Google Colab](#)

↓ [Descargar el cuaderno \(.ipynb\)](#)

Encadenamiento: un sistema experto que decide y explica

Qué prácticas: construir un motor de inferencia con encadenamiento hacia delante y hacia atrás, y que explique sus conclusiones. **Dónde encaja:** capítulo 4 (sistemas expertos y soporte a la decisión).

Construyes un sistema experto que decide si conceder un préstamo a partir de una base de reglas. Lo resuelves de las dos maneras y cuentas el esfuerzo: **hacia delante** (de los datos) deriva los hechos intermedios revisando las reglas hasta el punto fijo; **hacia atrás** (desde la meta «aprobar») sigue solo la cadena que la sostiene, con

muchas menos comprobaciones. Añades **factores de certeza** y propagas la confianza hasta la conclusión: se aprueba, pero con una certeza de alrededor del **53 %**, no del 100 %. Y el sistema **explica** la cadena completa de reglas y hechos.

► [Abrir en Google Colab](#)

↓ [Descargar el cuaderno \(.ipynb\)](#)

Razonamiento causal: de la correlación a la intervención

Qué prácticas: distinguir correlación de causa, ver la paradoja de Simpson y ajustar por un confusor. **Dónde encaja:** capítulo 5 (razonamiento causal).

Generas datos con un confusor y ves nacer una **correlación espuria**; reproduces la **paradoja de Simpson**, donde la tendencia agregada se invierte al estratificar; y calculas, sobre un modelo causal, la diferencia entre **observar** $P(Y | X)$ y **intervenir** $P(Y | do(X))$. Con la **fórmula de ajuste por puerta trasera** recuperas el efecto causal correcto y compruebas, con números, cuánto engañaba la correlación de partida.

► [Abrir en Google Colab](#)

↓ [Descargar el cuaderno \(.ipynb\)](#)

Razonamiento no monótono: la lógica que se retracta

Qué prácticas: negación por fallo, lógica por defecto con excepciones y revisión de creencias. **Dónde encaja:** capítulo 6 (razonamiento no monótono y revisión de creencias).

Construyes un mini-razonador que concluye «Tweety vuela» porque es un ave y **se retracta** al saber que es un pingüino: la no monotonía en acción. Implementas la **negación por fallo** (concluir «no vuela» cuando no se puede demostrar «vuela»), calculas la **extensión** de una base de reglas por defecto y montas una **revisión de creencias** al estilo AGM, donde la base incorpora un hecho que contradice lo anterior y restaura la consistencia quitando lo menos atrincherado. Acabas con un pequeño grafo de **argumentación**: argumentos que se atacan y cuáles quedan aceptados.

► [Abrir en Google Colab](#)

↓ [Descargar el cuaderno \(.ipynb\)](#)

En resumen

- La IA simbólica y el razonamiento bajo incertidumbre son **la otra mitad** del campo: lógica para lo tajante, probabilidad para lo incierto, difusa para lo gradual, sistemas expertos para orquestar con explicación y causalidad para distinguir correlación de causa.¹
- Estas herramientas no compiten con el aprendizaje profundo: lo **complementan**. Donde hace falta auditar, justificar, garantizar una regla o saber qué pasa al intervenir, lo simbólico y lo causal siguen ganando.
- Juntar las cinco sobre un mismo problema muestra la idea central del facsímil: **cada parte de una decisión merece la herramienta adecuada**, y un sistema que lo respeta es más robusto y más defendible que cualquier caja negra.

Para saber más

- Garcez, A. d'Avila, Lamb, L. C. y Gabbay, D. M. (2009). *Neural-Symbolic Cognitive Reasoning*. Springer.
- Jackson, P. (1998). *Introduction to Expert Systems* (3.^a ed.). Addison-Wesley.
- Pearl, J. y Mackenzie, D. (2018). *The Book of Why: The New Science of Cause and Effect*. Basic Books.
- Russell, S. y Norvig, P. (2021). *Artificial Intelligence: A Modern Approach* (4.^a ed.). Pearson.
- Zadeh, L. A. (1965). Fuzzy Sets. *Information and Control*, 8(3), 338-353.

Notas

1. Pearl, J. y Mackenzie, D. (2018). *The Book of Why*. Basic Books. Una defensa accesible de por qué la estadística sola no responde preguntas causales y hace falta el lenguaje de las intervenciones. ↵