Facsímil 12

Lógica e incertidumbre

Capítulo 04

Sistemas expertos y soporte a la decisión

686f6c61 · 2026

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.

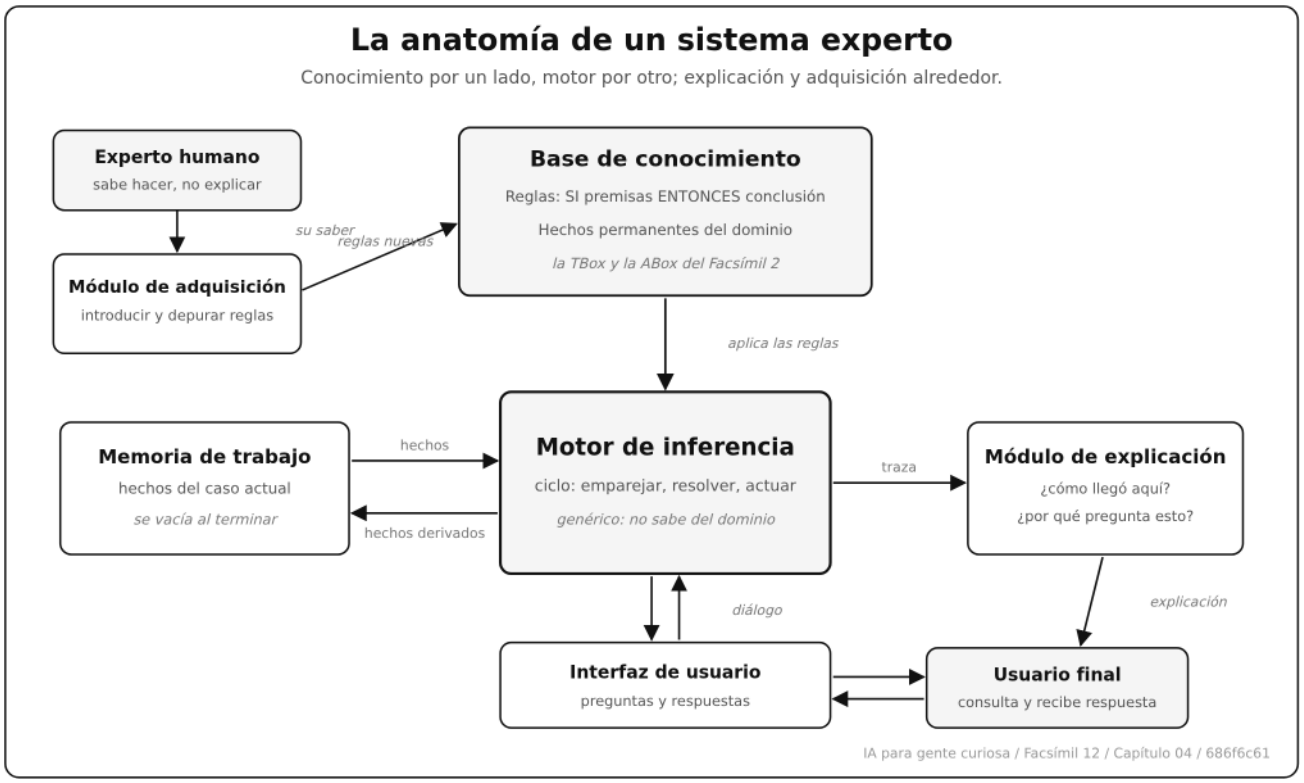
No es arqueología. Los sistemas expertos vivieron su edad de oro entre los años setenta y ochenta, pero su mecánica reaparece hoy, intacta, cada vez que un agente con un modelo de lenguaje consulta una tabla de permisos antes de ejecutar una acción, o cada vez que un *guardrail* aplica una regla de negocio 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.

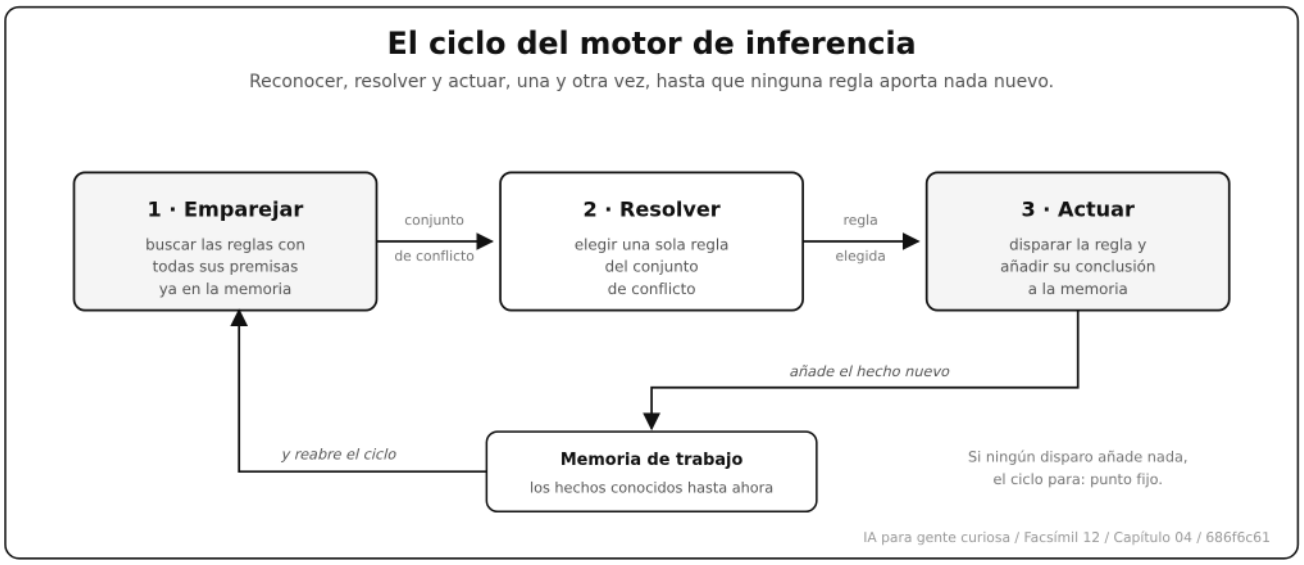
Antes de entrar en cada pieza, conviene verlas juntas. El siguiente diagrama coloca el motor de inferencia en el centro y dispone a su alrededor el resto de módulos: la base de conocimiento que lo alimenta, la memoria de trabajo con la que intercambia hechos, la explicación que produce y la adquisición 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	`nomina_domiciliada` ∧ `antiguedad_2anios`	`empleo_estable`
R2	`empleo_estable` ∧ `sin_impagos`	`solvencia_demostrada`
R3	`solvencia_demostrada` ∧ `ratio_deuda_bajo`	`riesgo_bajo`
R4	`riesgo_bajo` ∧ `importe_dentro_limite`	`aprobar`
R5	`ratio_deuda_alto`	`riesgo_alto`
R6	`riesgo_alto`	`escalar_a_analista`

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

$$\mathrm{MT}_0 = \{\texttt{nomina\_domiciliada}, \texttt{antiguedad\_2anios}, \texttt{sin\_impagos}, \texttt{ratio\_deuda\_bajo}, \texttt{importe\_dentro}$$

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 ∪ { 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	`empleo_estable`
2	R2	R2	`solvencia_demostrada`
3	R3	R3	`riesgo_bajo`
4	R4	R4	`aprobar`
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:

$$\mathrm{MT}_{\text{final}} = \mathrm{MT}_0 \cup \{\,\texttt{empleo\_estable},\ \texttt{solvencia\_demostrada},\ \texttt{riesgo\_bajo},\ \texttt{aprobar}\,\}$$

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.

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.

$$\mathrm{CF}(P_1 \wedge P_2 \wedge \cdots \wedge P_n) = \min_i \ \mathrm{CF}(P_i)$$

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.

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

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.

$$\mathrm{CF}(H) = \mathrm{CF}_1 + \mathrm{CF}_2 - \mathrm{CF}_1 \cdot \mathrm{CF}_2$$

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.

SÍMBOLO	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 × 0,88 × 0,88 × 0,88 × 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 $0{,}6 + 0{,}7 - 0{,}6 \times 0{,}7 =$ **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
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

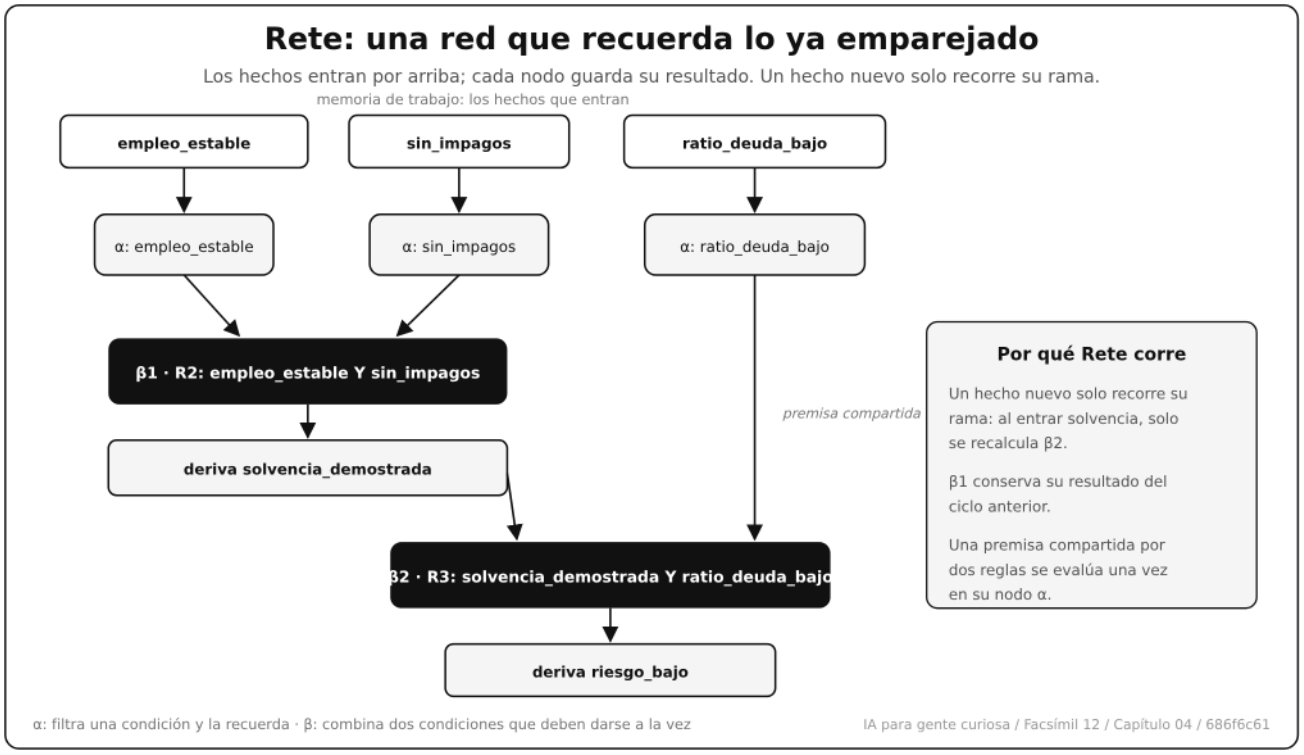
Hay un problema de eficiencia escondido en el algoritmo hacia delante. En cada ciclo recorríamos **todas** las reglas comprobando si sus premisas estaban en la memoria. Con seis reglas y cinco hechos no importa. Con diez mil reglas y cien mil hechos, reevaluarlo todo en cada ciclo es inviable: el motor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

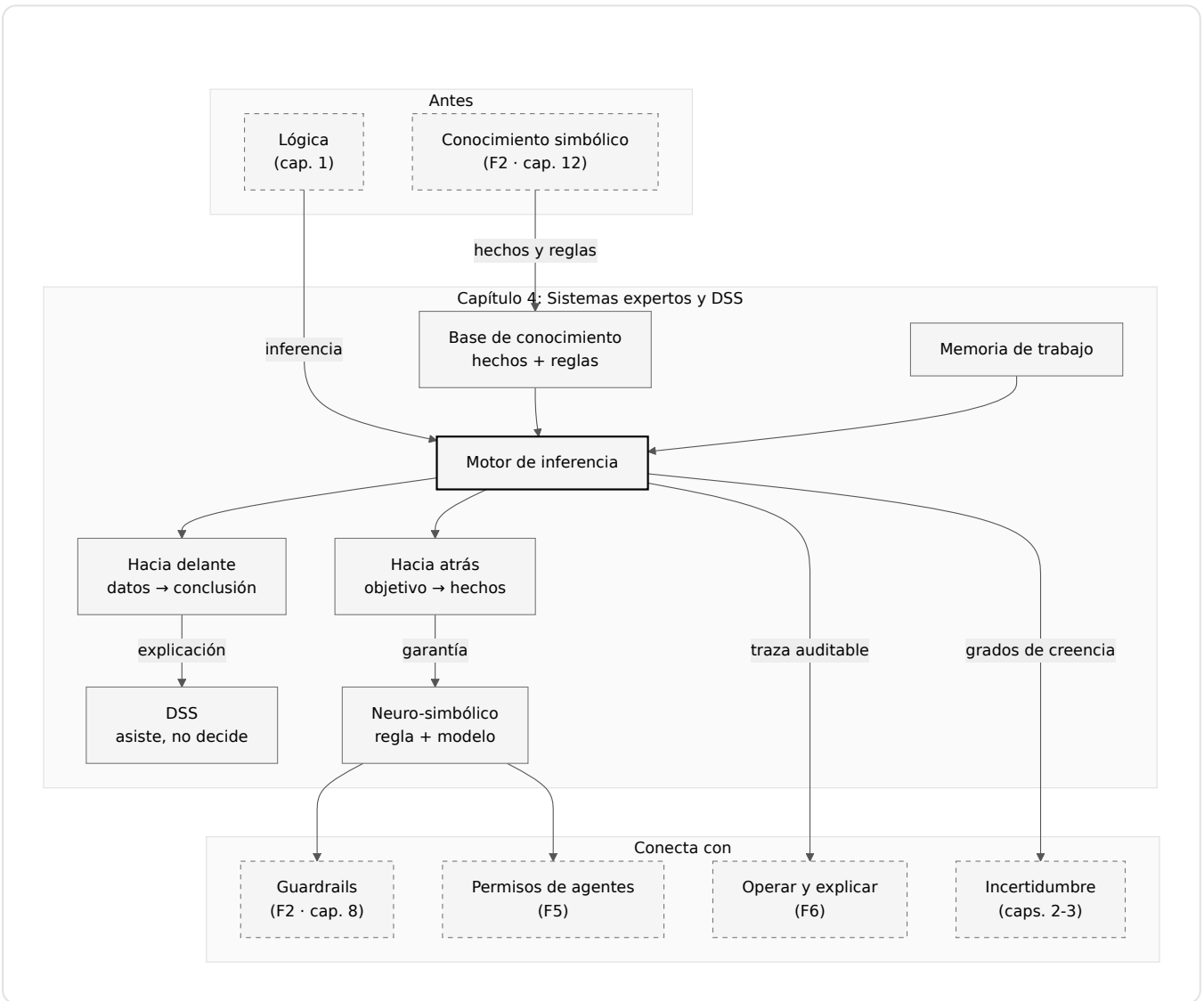
Ahora gira la vista hacia los agentes de hoy, porque es donde todo esto reaparece con ropa nueva. Los **permisos de un agente del Facsímil 5** son, sin más, un pequeño sistema experto. La tabla que decide si una herramienta está autorizada —«el rol analista no puede borrar sin confirmación», «por encima 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.

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

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 ↩

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