


Tema 5

Sistemas expertos.

En esta unidad veremos:

- 5.1 Introducción
- 5.2 Estructuras elementales de los sistemas expertos.
- 5.3 Representación y simulación de comportamientos básicos.
- 5.4 Estrategias de control de un sistema experto.
- 5.5 Tendencias en sistemas expertos.

Índice

Tema 5	1
5.1 Introducción.....	3
2. Estructuras elementales de los sistemas expertos.	3
5.2.1 Interfaz de usuario y de comunicación externa.	4
5.2.2 Base de datos de conocimiento.	4
La base de conocimientos en un sistema experto.	4
¿Cómo se construye esta base de conocimientos?	4
Tipos de conocimiento que puede contener un sistema experto	5
Ventajas de usar reglas del tipo SI...ENTONCES.....	5
Ejemplo de aplicación de un sistema experto: Concesión de créditos.....	5
Regla que implementaría el sistema experto	5
Conclusión	6
5.2.3 Motor de inferencias	6
4. Representación y simulación de comportamientos básicos.	8
Ejemplo 2: Código en PROLOG	9
Consulta específica por nivel.....	13
Consulta general de niveles	14
Interpretación	15
Relaciones entre personas: Ejemplo con “se llevan bien”	15
¿Qué significa esto?	15
4. Estrategias de control de un sistema experto.....	16
Encadenamiento hacia adelante	17
 Encadenamiento hacia atrás.....	17
Ejemplo 5: Identificación de animales	18
5. Tendencias en sistemas expertos.	18
Evolución: hacia sistemas expertos híbridos.....	19

5.1 Introducción.

Los **sistemas expertos** son una aplicación de la inteligencia artificial que utilizan **conocimientos especializados previamente adquiridos por seres humanos** en un campo determinado. Su desarrollo comenzó en la década de 1970 y alcanzaron gran popularidad durante esa época y en los años 80.

Se consideran los primeros sistemas de inteligencia artificial capaces de ofrecer resultados de utilidad práctica para los especialistas. Están basados, principalmente, en un conjunto de **reglas** que representan el conocimiento de expertos en un área determinada.

Para desarrollar un sistema experto, es imprescindible contar con el **conocimiento de un profesional** que domine el campo objeto de estudio. Es decir, se necesita información detallada sobre cómo resolvería un especialista un determinado problema.

Por ello, a menudo se les conoce también como **sistemas basados en conocimientos** o **sistemas basados en reglas**.

Un sistema experto puede definirse como un software diseñado para **simular el proceso de toma de decisiones de un experto humano** en un ámbito concreto. Está concebido para tomar decisiones de forma automática como si fuera un profesional. Además, es importante destacar que **todo sistema experto debe ser capaz de explicar las decisiones tomadas y de aprender** cuando se le proporciona nueva información.

2. Estructuras elementales de los sistemas expertos.

Un sistema experto está formado por varios componentes bien definidos que trabajan juntos para simular el razonamiento experto.

La arquitectura más común de los sistemas expertos es la del sistema basado en reglas. Este tipo de sistemas emplea expresiones del tipo: «SI ... ENTONCES».

Cada regla representa una porción del conocimiento que se pretende introducir en el sistema. Un conjunto de reglas relacionadas puede llevar de una serie de hechos y datos conocidos hasta algunas conclusiones de utilidad.

Todo sistema experto está formado por los siguientes elementos:

- Interfaz de usuario y de comunicación externa.
- Base de datos de conocimiento.
- Motor de inferencias.
- Sistema para la explicación de las decisiones tomadas.
- Sistema para la adquisición de nuevo conocimiento.

5.2.1 Interfaz de usuario y de comunicación externa.

En un sistema experto, **la interfaz de usuario** desempeña un papel fundamental, ya que permite una comunicación fluida entre el usuario y la aplicación. Esta interfaz puede estar basada en **texto, gráficos o una combinación de ambos**.

Dado que los sistemas expertos interactúan con el usuario durante el proceso de resolución de problemas, **es imprescindible que la interfaz facilite el usuario pueda realizar respuestas cómodas y comprensibles** a las preguntas planteadas por el sistema.

Asimismo, **las conclusiones generadas por el sistema se presentan al usuario a través de esta interfaz**, lo que refuerza su importancia.

Además, la interfaz debe contemplar una **sección para la comunicación externa**, ya que es habitual que el sistema necesite acceder a **datos externos** para completar su análisis o tomar decisiones informadas.

5.2.2 Base de datos de conocimiento.

La base de conocimientos en un sistema experto.

La **base de conocimientos** es el **núcleo fundamental** de todo sistema experto. Es donde se almacena toda la información relevante sobre el dominio que el sistema aborda. Este conocimiento está estructurado principalmente en dos tipos de elementos:

- **Hechos:** Información concreta sobre el dominio (por ejemplo: *"la planta tiene hojas amarillas"*).
- **Reglas:** Declaraciones del tipo **SI <condición> ENTONCES <acción>**, como por ejemplo:
SI la planta tiene hojas amarillas Y el suelo está seco → ENTONCES necesita riego.

¿Cómo se construye esta base de conocimientos?

Para su desarrollo, es necesario contar con la participación de **expertos humanos**, capaces de transformar su experiencia en **reglas lógicas**. Estas reglas pueden ser:

- **Simples:**
SI <condición> → ENTONCES <acción>
- **Compuestas:**
Usan conectores como **Y** u **O** para combinar múltiples condiciones:
 - *SI <condición 1> Y <condición 2> → ENTONCES <acción>*
 - *SI <condición 1> O <condición 2> → ENTONCES <acción>*

Tipos de conocimiento que puede contener un sistema experto

1. **Conocimiento procedimental:**
Describe **cómo realizar una tarea paso a paso**. Por ejemplo, un procedimiento para diagnosticar un fallo mecánico.
2. **Conocimiento factual:**
Es el saber **objetivo y documentado** que se puede encontrar en libros y manuales. Aunque accesible por otras vías, su inclusión en el sistema permite un acceso más rápido y eficaz.
3. **Conocimiento heurístico:**
Basado en la **experiencia y la intuición del experto**. No siempre es verificable en libros, pero es muy valioso para tomar decisiones en situaciones complejas o inciertas.

Ventajas de usar reglas del tipo SI...ENTONCES

- Son fáciles de entender, tanto para los programadores como para los usuarios.
- Permiten estructurar el conocimiento de forma clara y lógica.
- Facilitan el mantenimiento y la actualización del sistema.

Ejemplo de aplicación de un sistema experto: Concesión de créditos

Imaginemos que queremos diseñar un **sistema experto** que ayude a decidir si debe concederse un **crédito al consumo** a un cliente. Este sistema debe imitar el razonamiento que seguiría un empleado de banca con experiencia en este tipo de operaciones.

Un profesional del sector nos proporciona los **criterios clave**:

Solo se conceden créditos a personas:

- Mayores de 18 años
- Que tengan una **nómina**
- Y cuyo **contrato laboral** sea:
 - De tipo **indefinido**, o
 - De duración **superior al plazo de devolución del préstamo**

Regla que implementaría el sistema experto

Este conocimiento se puede transformar en una **regla lógica** del tipo:

SI

- el cliente es mayor de 18 años
Y tiene nómina
Y (tiene contrato indefinido

O el contrato tiene una duración superior al plazo del préstamo)
ENTONCES

- conceder el préstamo solicitado

Conclusión

Si este es el **único criterio** utilizado para decidir la concesión de créditos, un sistema experto que incorpore esta regla podrá **automatizar la decisión** de forma coherente, obteniendo resultados similares a los de un empleado humano con experiencia.

5.2.3 Motor de inferencias

Es el “cerebro” del sistema. El **motor de inferencias** es el componente encargado de **realizar el razonamiento lógico** dentro del sistema experto. A partir de los datos y reglas almacenadas en la base de conocimientos, este motor:

- Genera nueva información.
- Toma decisiones automáticas.
- Contribuye a resolver problemas reales imitando el proceso de pensamiento de un experto humano.

5.2.4 Sistema de explicación de decisiones

Una vez tomada una decisión, el sistema debe ser capaz de **explicarla de forma clara al usuario**. Esto se puede hacer mostrando:

- Las **reglas de inferencia** utilizadas en el razonamiento.

Sin embargo, este método puede ser poco práctico si es demasiado técnico o extenso. Por eso, los sistemas expertos deben contar con un **subsistema de explicación** que:

- **Traduzca el razonamiento técnico** a un lenguaje comprensible.
- **Justifique las decisiones** de manera clara y útil para el usuario final.

5.2.5 Sistema de adquisición de conocimiento

Para que un sistema experto se mantenga actualizado y siga siendo útil a lo largo del tiempo, debe contar con un **mecanismo que permita incorporar nuevo conocimiento**. Este subsistema debe:

- Servir de **interfaz para que los expertos humanos introduzcan nueva información**.
- Permitir la **actualización y ampliación constante** de la base de conocimientos.

Así, el sistema puede evolucionar y adaptarse a nuevas situaciones sin necesidad de ser reprogramado desde cero.

4. Representación y simulación de comportamientos básicos.

Ventajas de los sistemas expertos

Los sistemas expertos presentan diversas **ventajas clave**:

- **Fácil programación:**
La información se introduce mediante reglas del tipo **SI... ENTONCES**, lo que simplifica su implementación.
- **Gestión de gran cantidad de información:**
Son capaces de **manejar múltiples datos simultáneamente**, lo cual es ideal en entornos complejos.
- **Escalabilidad:**
Se pueden **ampliar fácilmente** añadiendo más reglas o información a la base de conocimientos.
- **Transparencia y valor pedagógico:**
La forma en que toman decisiones es comprensible y clara, lo cual **ayuda al aprendizaje** de quienes están empezando en una disciplina.

Limitaciones y desventajas

Pese a sus ventajas, los sistemas expertos también presentan algunas **limitaciones** importantes:

- **Cobertura incompleta:**
No pueden abarcar **todo el conocimiento** de una disciplina por muy amplia que sea la base de datos.
- **Falta de creatividad:**
No son capaces de **proponer soluciones innovadoras** o creativas como lo haría un ser humano.
- **Dependencia de la información programada:**
Solo pueden trabajar con los **datos y reglas previamente introducidos**. Un experto humano, en cambio, puede **investigar, contrastar o improvisar** antes de tomar una decisión.

Simulación con lenguaje PROLOG

Para comprender mejor los **comportamientos básicos de un sistema experto**, se presentan ejemplos en el lenguaje de programación **PROLOG** (acrónimo de *programmation et logique*).

- **Orígenes:**
Desarrollado en los años 70 por **Alain Colmerauer** y **Philippe Roussel** con el objetivo de crear un lenguaje cuya estructura se asemejara al **lenguaje natural**.

- **Estructura de un programa PROLOG:**

Se basa en una **base de datos** que contiene:

- **Hechos**
- **Reglas**

Estas se **evalúan** cuando se hace una consulta, dando como resultado **verdadero** o **falso**.

- **Lenguaje declarativo:**

A diferencia de los lenguajes imperativos (que indican cómo hacer algo paso a paso), **PROLOG describe lo que se quiere lograr**, basándose en **lógica formal**.

- **Ejemplo de uso:**

A través de PROLOG se puede preguntar si un hecho es verdadero o si una afirmación lógica es coherente con las reglas almacenadas.

- **Entorno de ejecución:**

Los ejemplos pueden probarse usando **SWI-Prolog**, un entorno de programación gratuito.

- **Lenguajes afines:**

Aunque hoy en día se puede programar un sistema experto en muchos lenguajes, **PROLOG y LISP** fueron creados específicamente para este tipo de aplicaciones.

Los sistemas expertos representan el conocimiento utilizando estructuras como:

- **Reglas de producción (IF-THEN).**
- **Redes semánticas o marcos (frames).**
- **Lógicas proposicionales o de predicados.**

Estas representaciones permiten simular comportamientos similares a los de un experto humano, como:

- Diagnosticar una enfermedad.
- Recomendar una acción concreta.
- Evaluar una situación y prever consecuencias.

Ejemplo 2: Código en PROLOG

La **Figura 5.1** muestra un fragmento de **código fuente en PROLOG** que representa un pequeño universo con:

- Personas definidas como individuos.
- Deportes disponibles.
- Clasificación de los deportes en tres categorías:
 - **Atletismo**
 - **Deportes de nieve**
 - **Deportes de equipo**

Un detalle que puede sorprender a quien no esté familiarizado con PROLOG es que los **nombres propios** (como los de las personas) aparecen **escritos con minúscula**. Esto no es un error, sino una **norma de la sintaxis del lenguaje**.

En PROLOG:

- Las **palabras que comienzan con minúscula** se interpretan como **constantes** (como los nombres de personas, lugares, objetos, etc.).
- Las **palabras que comienzan con mayúscula** se interpretan como **variables**.

Esta convención permite al lenguaje distinguir de forma automática entre elementos **fijos** (como “ana” o “tenis”) y elementos **variables** que se evaluarán en consultas o reglas.

```
% nombres de las personas persona (alfonso)
persona (ana) .
persona (carlos).
persona (carmen) •
persona (fernando).
persona (isabel) • persona (luis).
persona (maria).

% deportes y su clasificación tipo_deporte(marcha, atletismo).
tipo_deporte (vallas, atletismo)
tipo_deporte (cross, atletismo)
tipo_deporte (disco,atletismo).
tipo_deporte (martillo, atletismo).
tipo_deporte (salto,nieve).
tipo_deporte (esqui_alpino,nieve).
tipo_deporte (esqui_fondo, nieve)
tipo_deporte (snowboard, nieve).
tipo_deporte (trineo, nieve).
tipo_deporte (futbol, equipo).
tipo_deporte (baloncesto, equipo).
tipo_deporte (balonmano, equipo).
tipo_deporte (voleibol, equipo).
tipo_deporte (waterpolo, equipo).
```

Figura 5.1 Código fuente.

En la Figura 5.2 se presenta otro fragmento de código en el que se relaciona cada persona con el deporte que practica y, además, se indica a qué nivel lo hace. Así, Alfonso es un jugador de baloncesto que juega en la NBA, mientras que Ana es una saltadora de vallas que compite en categoría regional, Carlos es un atleta que se dedica a la marcha en competiciones de carácter nacional, Carmen juega al fútbol en un equipo de tercera, etc.

```
% deporte que practica cada uno y nivel
practica_deporte (alfonso, categoria (baloncesto, nba)
practica_deporte (ana, categoria (vallas, regional))
practica_deporte (carlos, categoria (marcha, nacional))
practica_deporte (carmen, categoria (futbol, tercera)
```

```
practica_deporte (fernando, categoria (baloncesto, aficionado))
practica_deporte (isabel, categoria (cross, regional))
practica_deporte (luis, categoria (snowboard, aficionado))
practica_deporte (maria, categoria (waterpolo, aficionado))
```

Figura 5.2 Código fuente

Una vez introducida la información en el sistema, es posible realizar consultas para recuperar los datos almacenados. Por ejemplo, se puede verificar si se han definido personas utilizando la consulta:

```
persona(Alguien).
```

En esta instrucción, la palabra `Alguien`, al comenzar con mayúscula, se interpreta como una **variable** en PROLOG. Esto significa que no estamos buscando una persona específica llamada “Alguien”, sino solicitando al sistema que nos devuelva cualquier persona registrada en su base de conocimientos.

En cambio, si se escribiera `persona(alguien) .`, con la primera letra en minúscula, PROLOG entendería que se busca una persona concreta cuyo nombre sea literalmente “alguien”.

Al ejecutar la consulta `persona(Alguien) .`, el sistema puede responder de dos formas:

- Si se pulsa **Enter** después del primer resultado, PROLOG muestra solo la **primera coincidencia** (por ejemplo, `Alguien = alfonso.`).
- Si se pulsa el **punto y coma (;)**, el sistema continuará **enumerando todas las personas definidas**, una por una.

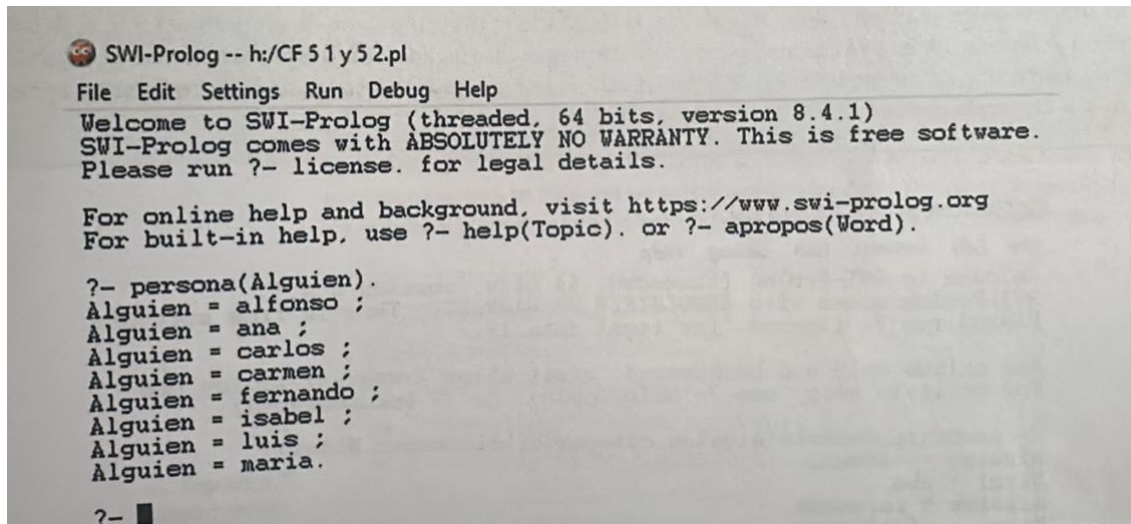
La **Figura 5.3** ilustra una captura de pantalla con la enumeración de resultados obtenidos mediante este tipo de consulta.

Por otro lado, si solo se desea saber si existe al menos una persona registrada, sin necesidad de conocer su nombre, se puede usar la consulta:

```
persona(_).
```

En este caso, el símbolo de subrayado (`_`) representa una **variable anónima**, lo que indica que no nos interesa el valor específico, sino solo comprobar la existencia del hecho. Si hay al menos una persona definida, el sistema devolverá simplemente: `true`.

Por último, es importante recordar que en PROLOG **todas las instrucciones y consultas deben terminar con un punto (.)**, ya que este símbolo actúa como terminador de comandos.

A screenshot of the SWI-Prolog IDE window. The title bar reads "SWI-Prolog -- h:/CF 5 1 y 5 2.pl". The menu bar includes "File", "Edit", "Settings", "Run", "Debug", and "Help". The main text area displays the following content:

```
Welcome to SWI-Prolog (threaded, 64 bits, version 8.4.1)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- persona(Alguien).
Alguien = alfonso ;
Alguien = ana ;
Alguien = carlos ;
Alguien = carmen ;
Alguien = fernando ;
Alguien = isabel ;
Alguien = luis ;
Alguien = maria.

?-
```

Figura 5.3 Captura de pantalla del resultado obtenido.

De forma similar al ejemplo anterior, si se desea conocer **qué deportes de nieve han sido definidos** en el sistema, puede utilizarse la siguiente consulta en PROLOG:

```
tipo_deporte(Nombres, nieve).
```

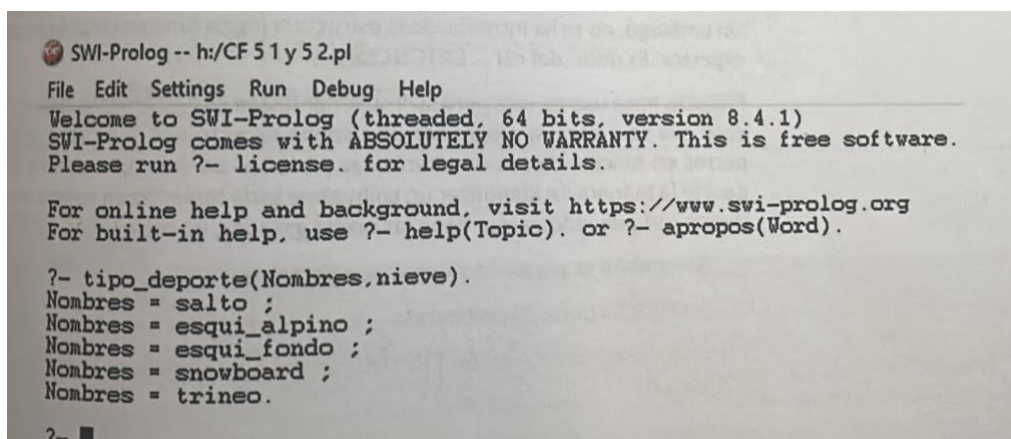
En esta instrucción:

- `Nombres` comienza con **mayúscula**, lo que indica que se trata de una **variable**. Por tanto, no se está creando un nuevo deporte llamado “Nombres”, sino solicitando al sistema que **devuelva todos los deportes** que pertenecen a la categoría **nieve**.

Al ejecutar esta consulta:

- El sistema mostrará el **primer resultado** (por ejemplo, `Nombres = esquí.`).
- Si se pulsa el **punto y coma (;)**, se obtendrán **todos los deportes de nieve definidos previamente** en la base de datos.

La **Figura 5.4** muestra una captura de pantalla con el resultado de esta consulta y la lista completa de deportes de nieve registrados.

A screenshot of the SWI-Prolog IDE window. The title bar reads "SWI-Prolog -- h:/CF 5 1 y 5 2.pl". The menu bar includes "File", "Edit", "Settings", "Run", "Debug", and "Help". The main text area displays the following content:

```
Welcome to SWI-Prolog (threaded, 64 bits, version 8.4.1)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- tipo_deporte(Nombres,nieve).
Nombres = salto ;
Nombres = esquí_alpino ;
Nombres = esquí_fondo ;
Nombres = snowboard ;
Nombres = trineo.

?-
```

Figura 5.4 Captura de pantalla del resultado obtenido.

Si queremos comprobar si en el universo definido por el código fuente **existe alguna persona que practique deporte**, podemos realizar la siguiente consulta en PROLOG:

```
practica_deporte(_, _).
```

El uso del **guion bajo** (`_`) indica que no nos interesa conocer los valores concretos, sino simplemente verificar si **existe al menos una coincidencia**. Es decir, esta consulta no busca nombres de personas ni deportes específicos, solo pregunta si **alguna persona practica algún deporte**.

- Si hay al menos un caso, el sistema responderá con: `true`.
- Al pulsar **punto y coma (;)** repetidamente, se seguirán obteniendo respuestas `true` por cada coincidencia encontrada, pero **no se mostrarán los nombres de las personas**.

Si se quiere saber **quién practica baloncesto y a qué nivel**, se puede emplear esta consulta:

```
practica_deporte(Alguien, categoria(baloncesto, Nivel)).
```

En este caso:

- `Alguien` es una variable que representa a la persona.
- `Nivel` es otra variable que devolverá el nivel de práctica.

Como resultado, el sistema puede responder, por ejemplo:

```
Alguien = alfonso,  
Nivel = nba ;  
Alguien = fernando,  
Nivel = aficionado.
```

Consulta específica por nivel

Si se desea saber **quién juega al baloncesto en la NBA**, la consulta específica sería:

```
practica_deporte(Alguien, categoria(baloncesto, nba)).
```

En este caso, el sistema devolverá únicamente: `Alguien = alfonso`.

Y al pulsar **punto y coma (;)**, responderá con: `false`.

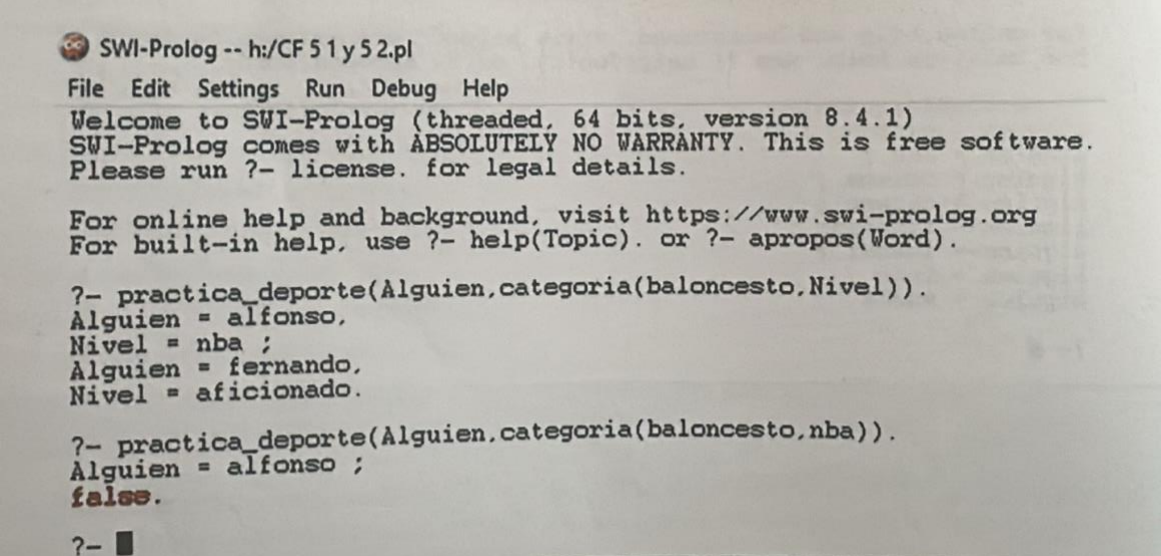
Esto indica que **no hay más personas** que practiquen baloncesto en la NBA.

Consulta general de niveles

Por último, si se desea conocer **el nivel al que practica su deporte cada persona**, puede usarse la consulta: `practica_nivel(Persona, Nivel)`.

Esta instrucción devolverá una lista con los nombres de las personas y el nivel correspondiente al que practican su deporte favorito.

Estos resultados se muestran en la **Figura 5.5**.



```
SWI-Prolog -- h:/CF 5 1 y 5 2.pl
File Edit Settings Run Debug Help
Welcome to SWI-Prolog (threaded, 64 bits, version 8.4.1)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- practica_deporte(Alguien,categoria(baloncesto,Nivel)).
Alguien = alfonso,
Nivel = nba ;
Alguien = fernando,
Nivel = aficionado.

?- practica_deporte(Alguien,categoria(baloncesto,nba)).
Alguien = alfonso ;
false.

?-
```

Figura 5.5 Captura de pantalla del resultado obtenido.

En el **Ejemplo 2** se ha utilizado el lenguaje de programación **PROLOG**, demostrando su capacidad para **razonar a partir de la información disponible**.

Sin embargo, hasta ahora no se ha explicado cómo se representa en PROLOG la **estructura lógica fundamental de los sistemas expertos**, es decir, las reglas del tipo “**SI... ENTONCES**”.

Aunque PROLOG utiliza este tipo de razonamiento, su **sintaxis es distinta** a la del lenguaje natural. Supongamos que queremos crear un sistema experto que clasifique **perros** en función de sus características físicas y su estilo de vida. Por ejemplo, para identificar a un **chihuahua**, podríamos enunciar la siguiente regla lógica:

SI el tamaño es pequeño **Y** las orejas son grandes,
ENTONCES el perro es un chihuahua.

En PROLOG, esta regla se expresa de la siguiente manera (ver Figura 5.6):


```
prolog  
  
perro(chihuahua) :- tamano(pequeno), orejas(grandes).
```

Figura 5.6

Interpretación

- El operador `:-` se traduce como “**SI**”.
- La **coma** `,` equivale al operador lógico “**Y**”.
- El operador “**;**” se utiliza para expresar la disyunción lógica, es decir, el “**O**”.

Por tanto, la línea de código anterior indica que:

"Un perro es de raza chihuahua si tiene un tamaño pequeño y orejas grandes."

Relaciones entre personas: Ejemplo con “se llevan bien”

En la **Figura 5.7**, se muestra otro fragmento de código en PROLOG que define una relación social: “**se_llevan_bien**”. Las siguientes reglas indican relaciones directas:

```
prolog  
  
se_llevan_bien(juan, pedro).  
se_llevan_bien(luis, ana).  
se_llevan_bien(maria, ana).  
  
se_llevan_bien(Cualquiera, pedro).  
se_llevan_bien(pedro, Cualquiera).
```

Figura 5.7 Código fuente

Además, se incluyen reglas más generales que utilizan una **variable**, para expresar relaciones universales.

¿Qué significa esto?

- La palabra **Cualquiera**, al comenzar con mayúscula, se interpreta como una **variable**.
- Estas reglas indican que:
 - **Cualquier persona se lleva bien con Pedro, y**
 - **Pedro se lleva bien con cualquier persona.**

Así, aunque no se haya definido explícitamente que Pedro y María se llevan bien, el sistema puede **deducir** esa relación gracias a la regla general. Por tanto, la consulta:

```
prolog  
  
se_llevan_bien(pedro, maria).
```

Debería devolver:

```
prolog  
  
true.
```

En cambio, si preguntamos por la relación entre Juan y María:

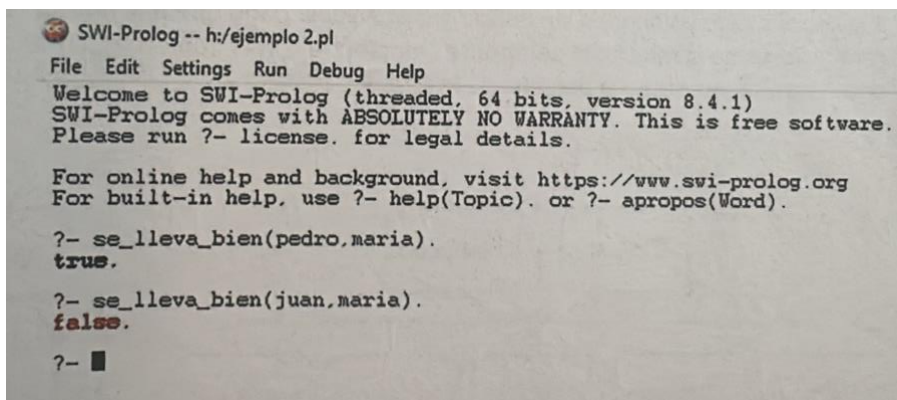
```
prolog  
  
se_llevan_bien(juan, maria).
```

El sistema responderá:

```
prolog  
  
false.
```

Porque **no existe una regla directa ni inferencia posible** en el código que indique esa relación.

La Figura 5.8 muestra una captura de pantalla de los resultados descritos en el párrafo anterior que confirma que las respuestas obtenidas fueron las esperadas.



```
SWI-Prolog -- h:/ejemplo 2.pl  
File Edit Settings Run Debug Help  
Welcome to SWI-Prolog (threaded, 64 bits, version 8.4.1)  
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.  
Please run ?- license. for legal details.  
  
For online help and background, visit https://www.swi-prolog.org  
For built-in help, use ?- help(Topic). or ?- apropos(Word).  
  
?- se_lleva_bien(pedro,maria).  
true.  
  
?- se_lleva_bien(juan,maria).  
false.  
  
?-
```

Figura 5.8 Captura de pantalla del resultado obtenido.

4. Estrategias de control de un sistema experto

En un sistema experto, el **motor de inferencia** puede aplicar distintas **estrategias de razonamiento** para llegar a una conclusión. Las dos más comunes son:

1. **Encadenamiento hacia adelante (razonamiento progresivo).**
2. **Encadenamiento hacia atrás (razonamiento regresivo).**

Ambas estrategias permiten obtener **resultados equivalentes**, pero la elección de una u otra depende del lenguaje de programación utilizado o del diseño específico del sistema.

- Por ejemplo, en el lenguaje **LISP**, se suele emplear el **encadenamiento hacia adelante**.
- En cambio, en **PROLOG**, es más habitual el uso del **encadenamiento hacia atrás**.

Encadenamiento hacia adelante

Esta estrategia parte de los **hechos conocidos** (las condiciones) y, aplicando reglas, **deduce nuevas conclusiones**. Funciona de manera natural y se asemeja al razonamiento humano.

Ejemplo 1:

```
SI el paciente tiene fiebre
Y ha perdido el olfato
ENTONCES hacer prueba PCR
```

Si ambas condiciones se cumplen, el sistema concluye que se debe realizar una prueba PCR.

Ejemplo 2:

```
SI llueve
ENTONCES usar paraguas
```

La presencia de lluvia activa automáticamente la recomendación de usar un paraguas.

Encadenamiento hacia atrás

Este enfoque parte de un **objetivo o hipótesis** y trata de verificar si se puede llegar a él mediante los datos disponibles y las reglas definidas. Es decir, se comienza por el resultado esperado y se va “tirando hacia atrás” para ver si las condiciones necesarias se cumplen.

Este tipo de razonamiento es típico de los **sistemas expertos implementados en PROLOG**, ya que se basa en comprobar si una conclusión puede justificarse a partir de reglas y hechos previos.

Ejemplo 5: Identificación de animales

Supongamos que queremos identificar una **mascota** a partir de ciertas observaciones. Sabemos que:

El animal **caza ratones** y **maulla** .

Contamos con las siguientes reglas en nuestra base de conocimiento:

1. SI un animal **caza ratones** y **maulla**,
ENTONCES **es un gato**.
2. SI un animal **pía** y **vuela**,
ENTONCES **es un pájaro**.
3. SI un animal **es un gato**,
ENTONCES **tiene pelo**.
4. SI un animal **es un pájaro**,
ENTONCES **tiene plumas**.

Aplicación:

Dado que el animal en cuestión **caza ratones** y **maulla**, aplicando la primera regla, se deduce que **es un gato**.

Luego, usando la tercera regla, también se puede afirmar que **tiene pelo**.

Este razonamiento muestra cómo, partiendo de una hipótesis (por ejemplo: *¿es un gato?*), el sistema puede **verificar o descartar** la conclusión usando las reglas y hechos almacenados.

5. Tendencias en sistemas expertos.

Aunque los sistemas expertos clásicos han sido en parte superados por el *machine learning*, hoy en día se integran en modelos híbridos con nuevas tendencias:

- **Sistemas expertos basados en lógica difusa**, que permiten trabajar con incertidumbre.
- **Sistemas expertos híbridos**, que combinan reglas con modelos de aprendizaje automático.
- **Explicabilidad en IA (XAI)**: enfoque moderno que retoma los sistemas expertos para justificar las decisiones de modelos complejos.
- **Uso en IoT y domótica**, con reglas personalizadas que controlan dispositivos inteligentes.
- **Integración con asistentes virtuales y chatbots**.

Principales beneficios

El uso de sistemas expertos aporta **ventajas significativas**:

- **Mejora en la calidad de las decisiones**
- **Reducción de costes operativos** y en la toma de decisiones
- **Mayor coherencia entre decisiones tomadas en diferentes momentos**
- **Agilidad en la respuesta y resolución de problemas**

Evolución: hacia sistemas expertos híbridos

En los últimos años, el desarrollo de sistemas expertos ha ido más allá de las tradicionales **reglas SI...ENTONCES**. Actualmente, se combinan con técnicas propias del **aprendizaje automático (machine learning)**.

¿Qué aportan estos sistemas híbridos?

- Son capaces de **aprender directamente de los datos**, sin depender exclusivamente del conocimiento de un experto.
- Esto permite que:
 - **Se reduzca el tiempo de desarrollo y puesta en marcha.**
 - **Sean más flexibles y adaptables.**
 - **Funcionen mejor en entornos con incertidumbre.**

En resumen, la tendencia actual se orienta hacia **sistemas expertos inteligentes y adaptativos**, que integran lo mejor del conocimiento humano estructurado y de la capacidad de aprendizaje de los algoritmos modernos.

Test evaluación

1. Los sistemas expertos son:

- a) Una metodología dentro del machine learning.
- b) Aplicaciones informáticas programas en LISP y orientadas a la toma de decisiones.
- c) Un tipo de sistemas de inteligencia artificial todavía en fase de desarrollo y para el que no se ha conseguido todavía ninguna aplicación práctica.
- d) Los sistemas expertos son una aplicación de la inteligencia artificial que hace uso de conocimientos especializados previamente adquiridos por el ser humano.

2. Los sistemas expertos se basan en:

- a) La implementación de reglas del tipo SI ... ENTONCES ...
- b) La programación y compilación de código LISP.
- c) El uso de redes neuronales artificiales.
- d) El uso de bases de datos heurísticas.

3. Todo sistema experto se compone de los siguientes elementos:

- a) Interfaz de usuario y de comunicación externa, base de datos de conocimiento, motor de inferencias, sistema para la explicación de las decisiones tomadas y sistema para la adquisición de nuevo conocimiento.
- b) Interfaz de usuario y de comunicación externa, base de datos de conocimiento, sistema para la explicación de las decisiones tomadas y sistema para la adquisición de nuevo conocimiento.
- c) Base de datos heurística y motor de inferencia procedimental.
- d) Ninguna de las respuestas anteriores es correcta.

4. El conocimiento que es propio de cada experto y que no se encuentra en los libros de texto se denomina:

- a) Conocimiento profundo.
- b) Conocimiento heurístico.
- c) Conocimiento factual.
- d) Conocimiento procedimental.

5. El motor de inferencias de un sistema experto:

- a) Es el elemento del sistema experto encargado de realizar el razonamiento.
- b) Es el elemento del sistema experto encargado de las búsquedas por internet.
- c) Es el elemento del sistema experto encargado de los cálculos en coma flotante.
- d) Es el elemento del sistema experto encargado de la programación de reglas heurísticas.