Representación del Conocimiento

Sistemas Expertos

Dr. Alejandro Guerra-Hernández

Instituto de Investigaciones en Inteligencia Artificial
Universidad Veracruzana

Campus Sur, Calle Paseo Lote II, Sección Segunda No 112,
Nuevo Xalapa, Xalapa, Ver., México 91097

mailto:aguerra@uv.mx
https://www.uv.mx/personal/aguerra/rc

Maestría en Inteligencia Artificial 2025

Universidad Veracruzana

Definición

- Se trata del primer producto de éxito de la IA.
- Un SE [2] es un programa que se comporta como un experto humano en algún dominio específico, resolviendo problemas mediante el conocimiento.
- Tareas comúnes:
  - Diagnóstico: Médico, fallo equipos, etc.
  - Interpretación de datos cuantitativos.
  - Toma de decisiones.

Universidad Veracruzana

Casos de estudio I

MYCIN. Shortliffe et al. [8] presentan un sistema para consultas en terapia anti-microbial. Posteriormente especializado en infecciones de la sangre y meningitis [9]. El libro de Buchanan y Shortliffe [3] presenta el sistema basado en reglas tras MYCIN.

DENDRAL. Lindsay et al. [5] proponen un sistema para deducir la estructura molecular de un compuesto químico a partir de datos obtenidos con un espectometro de masas. Lindsay et al. [6] ofrecen una panorámica de las dos décadas de desarrollo del sistema.

Universidad Veracruzana

Casos de estudio II

PROSPECTOR. Hart, Duda y Einaudi [4] proponen un sistema para la evaluación de sitios con posibles reservas minerales.

R1/XCON. Desarrollado por McDermott [7], es un sistema para la configuración de computadoras VAX-11/780, conforme a las ordenes de los clientes. El sistema fue renombrado como XCON y se uso en la compañía DEC [1].

Universidad Veracruzana

Desiderata

- Usa conocimiento para llevar a cabo sus tareas en el dominio de aplicación.
- Por ello se les conoce también como Sistemas Basados en el Conocimiento (KBS).
- Capaz de explicar sus decisiones y su manera de razonar.
- Capaz de contender con la incertidumbre y la falta de información inherentes a esta tarea.
- La explicación es útil para garantizar la confianza del usuario en las recomendaciones del sistema; o para detectar un fallo flagrante en su razonamiento.

Universidad Veracruzana

Incertidumbre e Incompletez

- Son inherentes a los SE. La información sobre el problema a resolver puede ser incompleta y/o no fiable.
- Las relaciones entre los objetos del dominio pueden ser aproximadas.
- Ejemplo. En el dominio médico, no estamos seguros si un síntoma se ha presentado en un paciente, o que la medida de un dato es absolutamente correcta: Ciertos fármacos pueden presentar reacciones adversas secundarias, pero éste no suele ser el caso.

Universidad Veracruzana

Funciones de un Sistema Experto

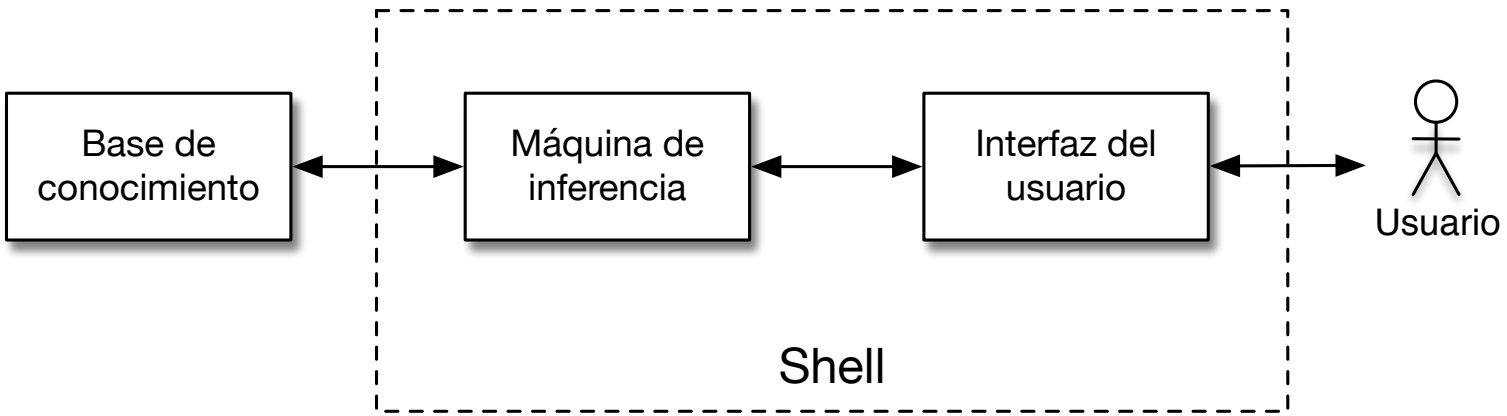
Solución del problema. Una función capaz de usar conocimiento sobre el dominio específico del sistema, incluyendo el manejo de incertidumbre.

Interfaz del usuario. Una función que permita la interacción entre el usuario y el sistema experto, incluyendo la capacidad de explicar las decisiones y el razonamiento del sistema.

Universidad Veracruzana

Arquitectura de un Sistema Experto

- Es conveniente dividir el SE en tres módulos, como se muestra a continuación:



Universidad Veracruzana

Módulos de un Sistema Experto I

Base de conocimientos. Todo el conocimiento específico sobre un dominio de aplicación: Hechos, reglas y/o restricciones que describen las relaciones en el dominio. Puede incluir también métodos, heurísticas e ideas para resolver los problemas en un dominio.

Máquina de inferencia. Todos los procedimientos para usar activamente la base de conocimientos.

Universidad Veracruzana

Módulos de un Sistema Experto II

Interfaz del usuario. Toda la comunicación entre el usuario y el sistema experto, debe proveer la información suficiente para que el usuario entienda el funcionamiento de la máquina de inferencia a lo largo del proceso.

Universidad Veracruzana

Shell de un Sistema Experto

- Conformado por la Máquina de inferencia y la interfaz del usuario.
- Es independiente del conocimiento del sistema.
- La separación provee modularidad.

Universidad Veracruzana

Reglas de producción

- Se les conoce también como reglas si-entonces.
- Son por mucho el formalismo de representación de conocimiento más utilizado en los SE.
- Pueden tener interpretaciones diversas:
  - Si condición P entonces conclusión C.
  - Si situación S entonces acción A.
  - Si las condiciones C_1 y C_2 son el caso, entonces la condición C no lo es.

Universidad Veracruzana

Características deseables

Modularidad. Cada regla define una pequeña, relativamente independiente, pieza de información.

Agregación. Es posible agregar nuevas reglas al sistema, de forma relativamente independiente al resto de sus reglas.

Flexibilidad. Como una consecuencia de la modularidad, las reglas del sistema pueden modificarse con relativa independencia de las otras reglas.

Transparencia. Facilitan la explicación de las decisiones tomadas por el sistema experto. Es posible automatizar la respuesta a preguntad del tipo ¿Cómo se llegó a esta conclusión? y ¿Porqué estás interesado en tal información?

Universidad Veracruzana

Tipos de conocimiento

Categórico. Se refiere a las reglas de producción que definen relaciones puramente lógicas entre conceptos del dominio de un problema. Son siempre absolutamente verdaderas.

Soft o Probabilístico. Prevalece en dominios como el médico, donde las relaciones no son siempre verdaderas y se establecen con base en su regularidad empírica, usando grados de certidumbre. Las reglas de producción deben ajustarse para contender con enunciados del tipo: Si condición A entonces conclusión C con un grado de certeza F.

Universidad Veracruzana

Ejemplo médico

- Aquí la regla se muestra en un pseudo-código más cercano a lo que haremos en Prolog:

```
Regla 037:
IF
paciente(hipertermia,si), paciente(tos,si), paciente(insufResp,si)
;
paciente(estertoresAlveolares,si), paciente(condensacionPulmonar,si)
THEN
paciente(neumonia,[si,80]).
```

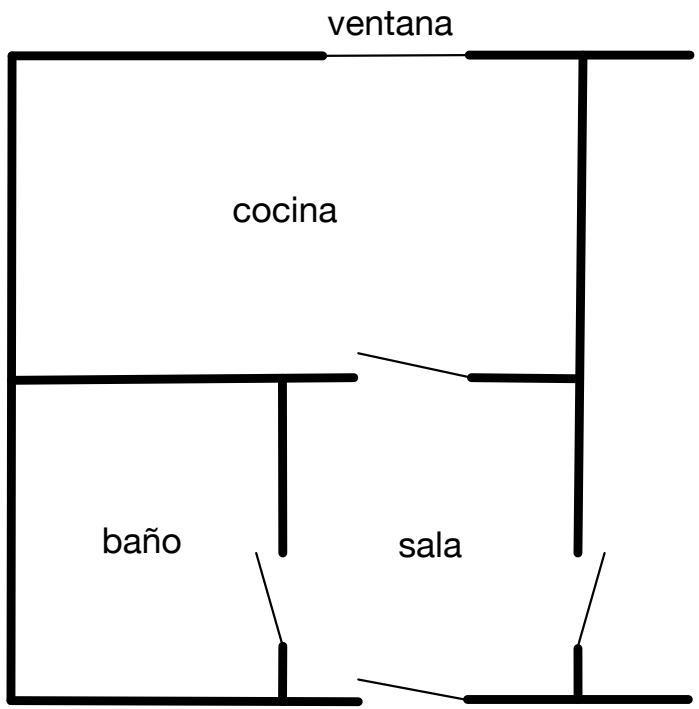
Universidad Veracruzana

Expertos humanos y validación

- Por lo general, si se quiere desarrollar un SE, será necesario consultar a un experto humano en el dominio del problema y estudiar el dominio de aplicación uno mismo.
- Al proceso de extraer conocimiento de los expertos y la literatura de un dominio dado, para acomodarlo a un formalismo de representación se le conoce como ingeniería del conocimiento (*knowledge elicitation*).
- El uso de aprendizaje automático no elimina del todo la interacción con los expertos humanos.

Universidad Veracruzana

Dominio para nuestros ejemplos [1]



Universidad Veracruzana

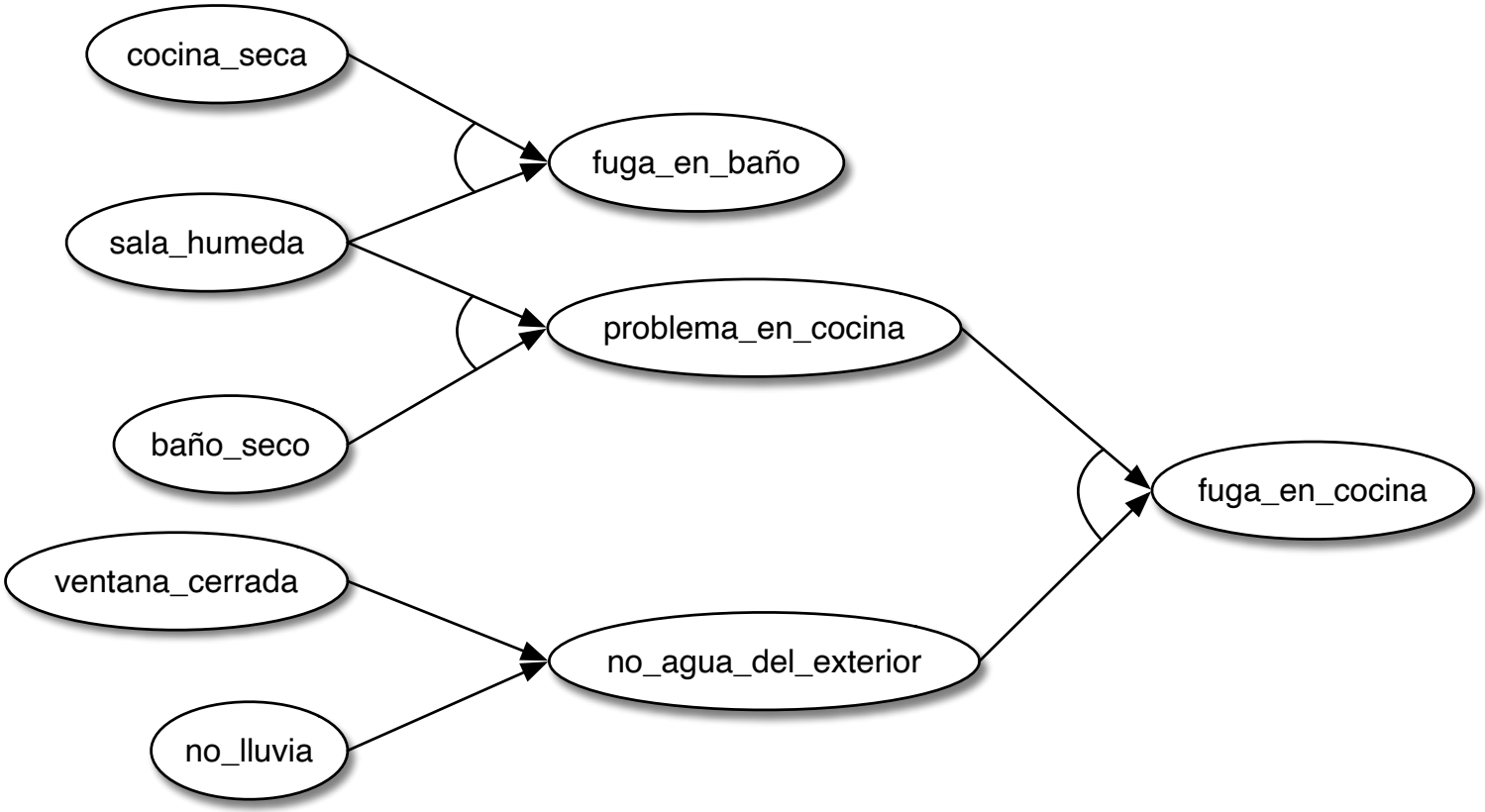
[1] Adaptado de Bratko [2].

Fugas de agua

- Luego de consultar con nuestro plomero favorito y consultar en internet los manuales de plomería para *dummies*, llegamos a la conclusión de que:
  - La fuga puede presentarse en el baño o en la cocina.
  - En cualquier caso, la fuga causa que haya agua en el piso de la sala.
  - Si la sala está humeda y el baño seco, entonces el problema está en la cocina.
  - Asumimos que la fuga solo se da en un sitio, no en ambos al mismo tiempo.
  - etc.
- ¿Cómo representamos este conocimiento?

Universidad Veracruzana

Red de inferencias (Gráfico AND/OR)



Universidad Veracruzana

Reglas

- Que se pueden representar como reglas de producción:
  `IF baño_seco AND sala_humeda THEN problema_en_cocina.`

Universidad Veracruzana

Tipos de razonamiento

- Hay dos formas básicas de razonamiento con reglas de producción:

  Hacía atrás. De la meta hacía los datos.
  Hacía adelante. De los datos hacía la meta.

Universidad Veracruzana

Razonamiento hacia atrás

- Partimos de una hipótesis: `fuga_en_cocina`; y razonamos hacia atrás sobre la red de inferencias, e.g., para resolver esta hipótesis necesitamos que `no_agua_en_el_exterior` y `problema_en_la_cocina` sean verdaderas.
- La primera es el caso si `no_lluvia` o `ventana_cerrada`, etc.
- El razonamiento se conoce como hacia atrás, porque forma una cadena de las hipótesis (`problema_en_cocina`) hacia las evidencias (`ventana_cerrada`).

Universidad Veracruzana

Implementación en Prolog

- ¡Por default!

Universidad Veracruzana

Reglas à la Prolog I

- La base de conocimientos:

```
1  % Base de conocimientos para fuga en casa
2
3  % Conocimiento previo
4
5  fuga_en_bano :-
6      sala_seca,
7      cocina_seca.
8
9  problema_en_cocina :-
10     sala_humeda,
11     bano_seco.
12
13 no_agua_del_exterior :-
14     ventana_cerrada
15     ;
16     no_lluvia.
17
18 fuga_en_cocina :-
19     problema_en_cocina,
```

Universidad Veracruzana

Reglas à la Prolog II

```
20      no_agua_del_exterior.
21
22  % Evidencia
23
24  sala_humeda.
25  bano_seco.
26  ventana_cerrada.
```

- La consulta para verificar nuestra hipótesis:

```
1  ?- fuga_en_cocina.
2  true
```

Universidad Veracruzana

Desventajas de usar Prolog directamente

1. La sintaxis de estas reglas no es la más adecuada para alguien que no conoce Prolog. Este es el caso de nuestros expertos humanos, que deben especificar nuevas reglas, leer las existentes y posiblemente modificarlas, sin saber Prolog.
2. La base de conocimientos es sintácticamente indistinguible del resto del sistema experto. Esto, como se mencionó, no es deseable.

Universidad Veracruzana

Intérprete de encadenamiento hacia atrás I

```
3   :-  op( 800, fx, if).
4   :-  op( 700, xfx, then).
5   :-  op( 300, xfy, or).
6   :-  op( 200, xfy, and).
7
8   is_true( P)  :-
9      fact( P).
10
11  is_true( P)  :-
12     if Cond then P,          % Una regla relevante,
13     is_true( Cond).          % cuya condición Cond es verdadera.
14
15  is_true( P1 and P2)  :-
16     is_true( P1),
17     is_true( P2).
18
19  is_true( P1 or P2)  :-
20     is_true( P1)
21     ;
22     is_true( P2).
```

Universidad Veracruzana

Nuevas reglas I

- La base de conocimientos incluye las reglas en el nuevo formato:

```
5  if sala_humeda and cocina_seca
6  then fuga_en_bano.
7
8  if sala_humeda and bano_seco
9  then problema_en_cocina.
10
11 if ventana_cerrada or no_lluvia
12 then no_agua_del_exterior.
13
14 if problema_en_cocina and no_agua_del_exterior
15 then fuga_en_cocina.
```

Universidad Veracruzana

Nuevas reglas II

- Y las evidencias del caso:

```
19  fact(sala_humeda).          % Cambiar a sala_seca para que falle la meta
20  fact(bano_seco).
21  fact(ventana_cerrada).      % Comentar para probar explicaciones porque
```

- La consulta es como sigue:

```
1  ?- is_true(fuga_en_cocina).
2  true
```

Universidad Veracruzana

Desventajas

- El usuario debe incluir en la base de conocimientos toda la evidencia con la que cuenta, en forma de hechos, antes de poder iniciar el proceso de razonamiento.

Universidad Veracruzana

Razonamiento hacia adelante

- Algunas veces resulta más natural razonar en la dirección opuesta: A partir de nuestra evidencia y nuestro conocimiento previo disponible, explorar que conclusiones podemos obtener.
- Ejemplo: Una vez que hemos confirmado que la sala está mojada y que el baño está seco, podríamos concluir que hay un problema en la cocina.

Universidad Veracruzana

Un interprete de razonamiento hacia adelante I

```
3  forward :-
4      new_derived_fact(P),     % Se deriva un nuevo hecho.
5      !,
6      write('Nuevo hecho derivado: '), write(P), nl,
7      assert(fact(P)),
8      forward   % Buscar más hechos nuevos.
9      ;
10     write('No se derivaron más hechos.').   % Terminar, no más hechos
       ↪  derivados.
11
12 new_derived_fact(Concl) :-
13     if Cond then Concl,    % Una regla
14     \+ fact(Concl),        % cuya conclusión no es un hecho
15     composed_fact(Cond).   % Su condición es verdadera?
16
17 composed_fact(Cond) :-
18     fact(Cond).            % Un hecho simple
19
20 composed_fact(Cond1 and Cond2) :-
21     composed_fact(Cond1),
22     composed_fact(Cond2). % Ambos operandos verdaderos.
```

Universidad Veracruzana

Un interprete de razonamiento hacia adelante II

```
23
24  composed_fact(Cond1 or Cond2) :-
25      composed_fact(Cond1)
26      ;
27      composed_fact(Cond2). % Al menos un operando verdadero.
```

Universidad Veracruzana

Consultas

- Se parte de la evidencia conocida, especificada mediante el predicado `fact`, se derivan los hechos que se pueden inferir mediante las reglas de producción y se agregan a la base de conocimiento, mediante el predicado `assert/1` (lo cual nos lleva a definir `fact/1` como dinámico).

```
?- forward.
Nuevo hecho derivado: problema_en_cocina
Nuevo hecho derivado: no_agua_del_exterior
Nuevo hecho derivado: fuga_en_cocina
No se derivaron más hechos.
true.
```

Universidad Veracruzana

Interfaz

- Una interfaz `loadSystem.pl` para cargar los módulos definidos hasta ahora, se implementa como sigue:

```
1  % Cargar el sistema basado en conocimiento.
2
3  :- dynamic(fact/1).
4
5  :- [encadenamientoAtras].
6  :- [encadenamientoAdelante].
7  :- [kbFugasIfThen].
```

Universidad Veracruzana

Comparando ambos razonamientos

- Las cadenas de inferencia conectan varios tipos de información como:

Datos	$\rightarrow \cdots \rightarrow$	Metas
Evidencia	$\rightarrow \cdots \rightarrow$	Hipótesis
Observaciones	$\rightarrow \cdots \rightarrow$	Diagnósticos
Descubrimientos	$\rightarrow \cdots \rightarrow$	Explicaciones
Manifestaciones	$\rightarrow \cdots \rightarrow$	Causas

Universidad Veracruzana

¿Qué tipo de razonamiento es mejor?

- Si lo que queremos es verificar si una hipótesis determinada es verdadera, entonces el razonamiento hacia atrás resulta más natural.
- Si es el caso que tenemos numerosas hipótesis y ninguna razón para comenzar con una de ellas en particular, entonces el encadenamiento hacia adelante es más adecuado.
- En general, el razonamiento hacia adelante es más adecuado en situaciones de monitoreo, donde los datos se adquieren continuamente.

Universidad Veracruzana

Red de inferencias revisitada

- Si solo hay unos pocos nodos de datos (el flanco izquierdo de la gráfica) y muchos nodos meta (el flanco derecho), entonces el encadenamiento hacia adelante sería el más apropiado;
- y viceversa.

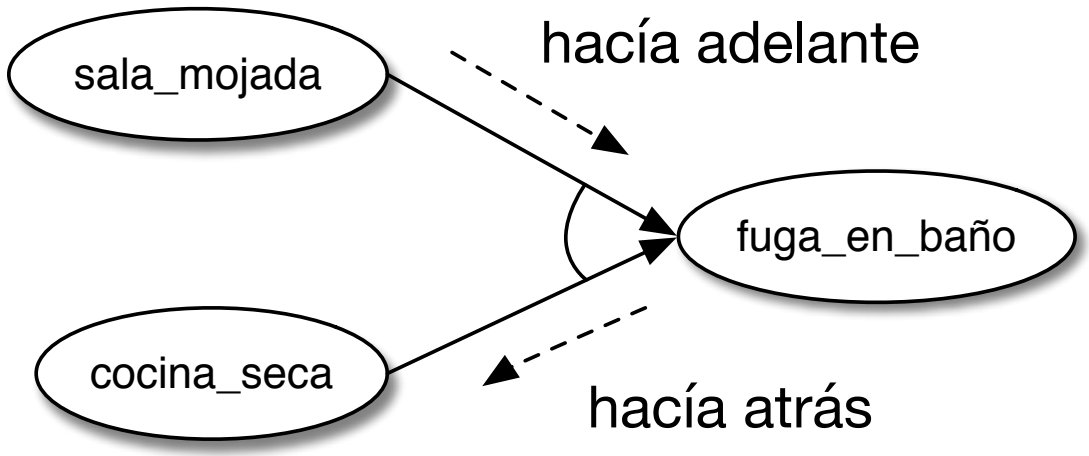
Universidad Veracruzana

Razonamiento bi-direccional

- En medicina, por ejemplo, normalmente algunas observaciones iniciales disparan razonamientos del médico hacia adelante, para generar una hipótesis de diagnóstico inicial.
- Ésta debe ser confirmada o rechazada con base en evidencia adicional, la cual se obtiene mediante procesos de razonamiento hacia atrás.

Universidad Veracruzana

Ejemplo en nuestro dominio



Universidad Veracruzana

Tipos de explicación

- Existen formas estándar de generar explicaciones en los sistemas basados en reglas de producción.
- Los tipos de explicación computables son respuestas a las preguntas de tipo:
  - ¿Cómo sabes esto?
  - ¿Porqué necesitas saber esto?

Universidad Veracruzana

Ejemplo de explicación tipo ¿Cómo?

- Supongamos que la última respuesta computada por nuestro sistema es que hay una fuga en la cocina. La explicación de esta respuesta sería como sigue:
  1. Hay un problema en la cocina, lo cual fue concluído a partir de que la sala está mojada y el baño seco.
  2. El agua no provino del exterior, lo cual fue concluído a partir de que la ventana estaba cerrada.
- Tal explicación es de hecho el árbol de derivación-SLD de la meta!

Universidad Veracruzana

Algoritmo para explicaciones ¿Cómo sabes esto?

- Definamos <= como un operador infijo, para poder representar el árbol de prueba de una proposición P de la siguiente manera:

  1. Si P es un hecho, entonces el árbol de prueba es P mismo.
  2. Si P fue derivado usando una regla: if *Cond* then P, el árbol de prueba es P <= *PruebaCond*; donde *PruebaCond* es a su vez, el árbol de prueba de *Cond*.
  3. Sean P_1 y P_2 proposiciones cuyos árboles de prueba son A_1 y A_2, entonces el árbol de prueba de P_1 `and` P_2 es A_1 `and` A_2. El caso de `or`, se resuelve análogamente.

Universidad Veracruzana

Implementación I

```
1  % is_true(P,A) A es el árbol de prueba de la meta proposicional P
2
3  :- op(800, xfx, <=).
4
5  is_true(P,P) :-
6      fact(P).
7
8  is_true(P, P <= CondA) :-
9      if Cond then P,
10     is_true(Cond,CondA).
11
12 is_true(P1 and P2, A1 and A2) :-
13     is_true(P1,A1),
14     is_true(P2,A2).
15
16 is_true(P1 or P2, A1 or A2) :-
17     is_true(P1,A1)
18     ;
19     is_true(P2,A2).
```

Universidad Veracruzana

Interfaz V2

```
1  % Cargar el sistema basado en conocimiento.
2
3  :- dynamic(fact/1).
4
5  :- [encadenamientoAtras].
6  :- [encadenamientoAdelante].
7  :- [explicacionesComo].
8  :- [kbFugasIfThen].
```

Universidad Veracruzana

Ejemplo: Consulta con explicaciones ¿Cómo?

```
?- [loadSystemV2].
true.

?- is_true(fuga_en_cocina,A).  A =
(fuga_en_cocina<=(problema_en_cocina<=sala_humeda and
bano_seco)and(no_agua_del_exterior<=ventana_cerrada or _))
```

Universidad Veracruzana

Formatear la explicación I

```
21  % show_tree(A): Imprime el árbol de derivación A.
22
23  show_tree(A) :-
24      show_tree(A,0).
25
26  % show_tree(A,N): Imprime el árbol de derivación A,
27  % al nivel de indentación N.
28
29  show_tree(A,N) :-
30      var(A), tab(N), writeln('cualquier_cosa').
31
32  show_tree(A1 <= A2,N) :- !,
33      tab(N), write(A1), writeln(' <='),
34      show_tree(A2,N).
35
36  show_tree(A1 and A2,N) :- !,
37      N1 is N + 2,
38      show_tree(A1,N1), tab(N1), writeln('and'),
39      show_tree(A2,N1).
40
41  show_tree(A1 or A2,N) :- !,
42      N1 is N + 2,
```

Universidad Veracruzana

Formatear la explicación II

```
43        show_tree(A1,N1), tab(N1), writeln('or'),
44        show_tree(A2,N1).
45
46    show_tree(A,N) :-
47        tab(N),writeln(A).
```

Universidad Veracruzana

Consulta mejorada

```
?- [loadSystemV2].
true.

?- is_true(fuga_en_cocina,A), show_tree(A).
fuga_en_cocina <=
  problema_en_cocina <=
    sala_humeda
    and
    bano_seco
  and
  no_agua_del_exterior <=
    ventana_cerrada
    or
    cualquier_cosa

A = (fuga_en_cocina<=(problema_en_cocina<=sala_humeda and
↪  bano_seco)and(no_agua_del_exterior<=ventana_cerrada or _))
```

Universidad Veracruzana

Explicaciones del tipo ¿Porqué necesitas saber esto?

- Se necesitan en tiempo de ejecución, no al final de una consulta.
- Requieren que el usuario pueda interactuar con la máquina de inferencia del sistema experto, através de la interfaz.
- Será necesario especificar que proposiciones pueden ser preguntadas al usuario, mediante el predicado `askable/1`, agregando a la base de conocimientos:

```
23 %%% Hechos que se pueden preguntar la usuario
24
25 askable(no_lluvia).
26 askable(ventana_cerrada).
```

Universidad Veracruzana

Algoritmo

- Dado `askable(P)`, cada vez que la meta `P` se presente, el sistema preguntará si `P` es el caso, con tres respuestas posibles: `si`, `no`, y `porque`.
- En caso de respuesta `si` al sistema, la proposición `P` es agregada a la base de conocimientos, usando `assert(fact(P))`.
- Debemos agregar también una cláusula al estilo de `already_asked(P)`, para evitar preguntar de nuevo por `P` cuando esto ya no es necesario.
- La respuesta a `porque` debería ser una cadena de reglas que conectan la evidencia `P` con la meta original

Universidad Veracruzana

Implementación de explicaciones tipo ¿Porqué? I

```
1  :- dynamic already_asked/1.
2
3  % iis_true(P,A): P es el caso con la explicación A.
4  % Versión interactiva con el usuario.
5
6  iis_true(P,A) :-
7      explore(P,A,[]).
8
9  % explore(P,A,T): A es una explicación de porque P es verdadera, T
10 % es una cadena de reglas que liga P con las metas anteriores.
11
12 explore(P, P, _) :-
13     fact(P).
14
15 explore(P1 and P2, A1 and A2, T) :-
16     explore(P1, A1, T),
17     explore(P2, A2, T).
18
19 explore(P1 or P2, A1 or A2, T) :-
20     explore(P1, A1, T)
21     ;
22     explore(P2, A2, T).
```

Universidad Veracruzana

Implementación de explicaciones tipo ¿Porqué? II

```
23
24  explore(P, P <= ACond, T) :-
25      if Cond then P,
26      explore(Cond, ACond, [if Cond then P | T]).
27
28  explore(P,A,T) :-
29      askable(P),
30      \+ fact(P),
31      \+ already_asked(P),
32      ask_user(P, A, T).
33
34  % ask_user(P,A,T): Pregunta al usuario si P es el caso,
35  % generando las explicaciones A y T.
36
37  ask_user(P, A, T) :-
38      nl, write('¿Es cierto que: '), write(P), write('?'),
39      writeln(' Conteste si/no/porque:'),
40      read(Resp),
41      process_answer(Resp,P,A,T).
42
43  process_answer(si,P, P <= preguntado,_) :-
```

Universidad Veracruzana

Implementación de explicaciones tipo ¿Porqué? III

```
44      asserta(fact(P)),
45      asserta(already_asked(P)).
46
47  process_answer(no,P,_,_) :-
48      asserta(already_asked(P)),
49      fail.
50
51  process_answer(porque,P,A,T) :-
52      display_rule_chain(T,0), nl,
53      ask_user(P,A,T).
```

Universidad Veracruzana

Formato de la explicación I

```
55  % display_rule_chain(R,N): Despliega las reglas R,
56  % identando a nivel N.
57
58  display_rule_chain([],_).
59
60  display_rule_chain([if Cond then P | Reglas], N) :-
61      nl, tab(N), write('Para explorar si '),
62      write(P), write(' es el caso, usando la regla:'),
63      nl, tab(N), write(if Cond then P),
64      N1 is N + 2,
65      display_rule_chain(Reglas,N1).
```

Universidad Veracruzana

Interfaz V3

```
1  % Cargar el sistema basado en conocimiento.
2
3  :- dynamic(fact/1).
4
5  :- [encadenamientoAtras].
6  :- [encadenamientoAdelante].
7  :- [explicacionesComo].
8  :- [explicacionesPorque].
9  :- [kbFugasIfThen].
```

Universidad Veracruzana

Observaciones

- El predicado principal se llama ahora `iis_true/2`, para indicar que es la versión interactiva del anterior `is_true/2`
- De otra forma hay un conflicto de nombres, que Prolog resuelve definiendo nuevamente el predicado en cuestión a partir del último archivo cargado.

Universidad Veracruzana

Preliminares a la consulta

- Antes de ejecutar la consulta, verifique que ha comentado la línea correspondiente a `fact(ventana_cerrada)` en la base de conocimientos original, para que el sistema experto deba preguntar por este hecho o por `no_lluvia`.
- El usuario puede pedir explicaciones al respecto en ambos casos.

Universidad Veracruzana

Ejemplo de consulta con explicación ¿Porqué? I

```
?- [loadSystemV3].
true.

?- iis_true(fuga_en_cocina,A), show_tree(A).

¿Es cierto que: ventana_cerrada? Conteste si/no/porque:
|: porque.

Para explorar si no_agua_del_exterior es el caso, usando la regla:
if ventana_cerrada or no_lluvia then no_agua_del_exterior
  Para explorar si fuga_en_cocina es el caso, usando la regla:
  if problema_en_cocina and no_agua_del_exterior then fuga_en_cocina

Es cierto que: ventana_cerrada Conteste si/no/porque:
|: si.

fuga_en_cocina <=
  problema_en_cocina <=
    sala_humeda
    and
    bano_seco
```

Universidad Veracruzana

Ejemplo de consulta con explicación ¿Porqué? II

```
22      and
23      no_agua_del_exterior <=
24        ventana_cerrada
25        or
26        cualquier_cosa
27
28  A = (fuga_en_cocina<=(problema_en_cocina<=sala_humeda and
29        bano_seco)and(no_agua_del_exterior<=(ventana_cerrada<=preguntado)or
          ↪  _))
```

Universidad Veracruzana

Limitaciones

- Obviamente el módulo de explicaciones del tipo ¿Porqué? solo funciona con reglas proposicionales.
- Si usásemos variables, habría que preguntar también por los valores de estas.
- No verificamos que las respuestas del usuario sean válidas. Ver el tutorial de prolog E/S.

Universidad Veracruzana

Grados de certeza

- Mucha de la experticia humana no es categórica. Tanto los datos como las reglas asociadas a un dominio de problema, pueden ser parcialmente ciertos.
- Esta incertidumbre se puede modelar si asignamos alguna calificación, que no sea verdadero o falso, a los hechos.
- Ejemplo: Un valor entre 0 y 1.
- Tales valores reciben diversos nombres como grados de certeza, o grados de creencia, o probabilidad subjetiva.
- No están basados estrictamente en la teoría de probabilidad.

Universidad Veracruzana

Esquema de certeza

- Cada proposición P será asociada a un número FC entre 0 y 1, que representará su factor de certeza.
- Usaremos un par $P : FC$ para esta representación. La notación adoptada para las nuevas reglas será por lo tanto:

$$\texttt{if } \mathit{Cond} \texttt{ then } P : FC.$$

Universidad Veracruzana

Combinación de factores de certeza

- El esquema de combinación será el siguiente

$$c(P_1 \textit{ and } P_2) = \text{mín}(c(P_1), c(P_2)) \quad (1)$$
$$c(P_1 \textit{ or } P_2) = \text{máx}(c(P_1), c(P_2)) \quad (2)$$

- En el caso de una regla `if` P_1 `then` $P_2 : FC$, entonces:

$$c(P_2) = c(P_1) \times FC \quad (3)$$

Universidad Veracruzana

Simplificaciones

- Por simplicidad asumiremos que las reglas tienen implicaciones únicas.
- Si ese no fuese el caso, las implicaciones pueden escribirse como disyunciones para satisfacer esta restricción de formato.

Universidad Veracruzana

Nueva base de conocimientos I

```
%%% base de conocimiento para fugas, en fomato de reglas if-then

%%% Conocimiento previo con certidumbre

if sala_humeda and cocina_seca
then fuga_en_bano.

if sala_humeda and bano_seco
then problema_en_cocina:0.9.

if ventana_cerrada or no_lluvia
then no_agua_del_exterior.

if problema_en_cocina and no_agua_del_exterior
then fuga_en_cocina.

% Evidencia con certidumbre

given(sala_humeda, 1).      % verdadero
given(bano_seco, 1).
given(cocina_seca, 0).      % falso
given(no_lluvia, 0.8).      % muy probable
```

Universidad Veracruzana

Nueva base de conocimientos II

```
23  given(ventana_cerrada, 0).
```

Universidad Veracruzana

Implementación I

```
1  % Interprete basado en reglas con incertidumbre
2
3  % cert(P,C): C es el grado de certeza de la proposición P
4
5  cert(P,C) :-
6      given(P,C).
7
8  cert(P1 and P2, C) :-
9      cert(P1,C1),
10     cert(P2,C2),
11     min(C1,C2,C).
12
13 cert(P1 or P2, C) :-
14     cert(P1,C1),
15     cert(P2,C2),
16     max(C1,C2,C).
17
18 cert(P, C) :-
19     if Cond then P:C1,
20     cert(Cond,C2),
21     C is C1 * C2.
22
```

Universidad Veracruzana

Implementación II

```
23  cert(P, C) :-
24      if Cond then P,
25      cert(Cond,C1),
26      C is C1.
27
28  % max y min
29
30  max(X,Y,X) :- X>=Y,!.
31  max(_,Y,Y).
32
33  min(X,Y,X) :- X=<Y,!.
34  min(_,Y,Y).
```

Universidad Veracruzana

Interfaz V4

```
1  % Cargar el sistema basado en conocimiento.
2
3  :- dynamic(fact/1).
4
5  :- [encadenamientoAtras].
6  :- [encadenamientoAdelante].
7  :- [incertidumbre].
8  :- [kbFugasIncert].
```

Universidad Veracruzana

Consulta con incertidumbre

```
?- cert(fuga_en_cocina,FC).
FC = 0.8
```

Universidad Veracruzana

Limitaciones

- Los factores de certeza no son probabilidades, en el sentido estricto del término.
- Asumamos que:
  - el factor de certeza de a es 0.5 y el de b es 0.
  - Entonces, el factor de certeza de a `or` b es 0.5.
  - Ahora, si en factor de certeza de b se incrementa a 0.5, esto no tiene efecto en el factor de certeza de la disyunción, que sigue siendo 0.5.

  lo cual resulta, al menos, contra intuitivo.

Universidad Veracruzana

Objeciones transnochadas

- Los expertos humanos parecen tener problemas para razonar basados realmente en la teoría de probabilidad. La verosimilitud que usan, no se corresponde con la noción de probabilidad definida matemáticamente.
- El procesamiento con base en la teoría de probabilidad, parece requerir de información que no siempre está disponible; o asumir simplificaciones que no siempre se justifican.

Universidad Veracruzana

Redes Bayesianas

- También se les conoce como Redes de creencias [2].
- Usan el cálculo de probabilidades para manejar la incertidumbre inherente a las representaciones de conocimiento.
- Proveen mecanismos eficientes para manejar las dependencias probabilísticas, mientras explotan las independencias;
- Resultando en una representación "natural" de la causalidad.

Universidad Veracruzana

Supuestos

- El estado del medio ambiente puede representarse por medio de un vector de variables.
- Nuestras variables tienen un dominio booleano, es decir, podrán tener como valor falso o verdadero.
- Ejemplo: `ladron` y `alarma`.
- Se puede generalizar a dominios no booleanos, pero discretos; o continuos con ciertas restricciones.

Universidad Veracruzana

Eventos

- Cuando las variables son booleanas, es normal pensar en ellas como si representasen eventos.
- Ejemplo: Si la alarma suena en el mundo real, es de esperar que la variable `alarma` se vuelve verdadera.

Universidad Veracruzana

Probabilidades

- Un agente, natural o artificial, normalmente no está completamente seguro de cuando estos eventos son verdaderos o falsos.
- El agente razona acerca de la probabilidad de que la variable en cuestión sea verdadera.
- Las probabilidades en este caso, son usadas como una medida del grado de creencia.
- Este grado depende de qué tanto el agente conoce su medio ambiente.
- Tales creencias se conocen tambien como probabilidades subjetivas, que no arbitrarias.

Universidad Veracruzana

Notación

- Sean X e Y dos variables, entonces, como de costumbre:
  - $X \wedge Y$ denota la conjunción de X e Y.
  - $X \vee Y$ denota la disyunción de X e Y.
  - $\neg X$ denota la negación de X.
- La expresión $p(X)$ denota la probabilidad de que la proposición X sea verdadera.
- La expresión $p(X \mid Y)$ denota la probabilidad condicional de que la proposición X sea verdadera, dado que la proposición Y lo es.

Universidad Veracruzana

Nuevas metas

- Un meta típica es este contexto toma esta forma: Dado que los valores de ciertas variables han sido observados ¿Cuales son las probabilidades de que otras variables de interés tomen cierto valor específico?
- O, dado que ciertos eventos han sido observados ¿Cual es la probabilidad de que sucedan otros eventos de interés?
- Ejemplo: Si observamos que la alarma suena ¿Cual es la probabilidad de que un ladrón haya entrado en la casa?

Universidad Veracruzana

Problemas

- La mayor dificultad para resolver estas metas está en manejar la dependencia entre las variables del problema.
- Si nuestro problema contempla n variables booleanas, necesitamos $2^n - 1$ números para definir la distribución de probabilidad completa entre los 2^n estados posibles del medio ambiente!
- Esto no sólo es costoso y nada práctico, sino que suele ser imposible, dada la disposición de esta información.
- Afortunadamente, suele ser el caso que no es necesario contar con todas esas probabilidades.

Universidad Veracruzana

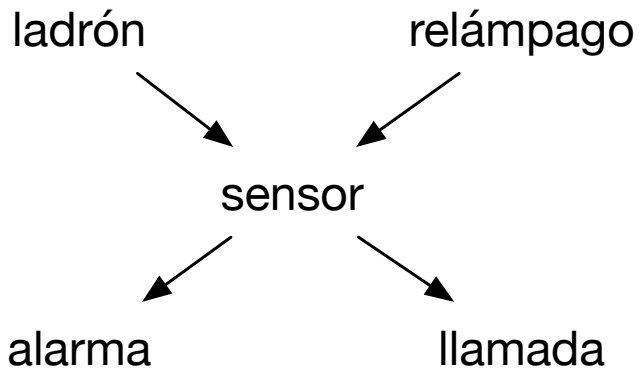
Dependencia, independencia y redes bayesianas

- La distribución de probabilidades completa, no asume nada acerca de la independencia entre variables.
- Afortunadamente, algunos eventos pueden ser independientes de otros.
- Las redes bayesianas proveen una forma elegante para declarar la dependencia e independencia entre variables.

Universidad Veracruzana

Ejemplo de red bayesiana

- La estructura de esta red indica dependencias e independencias probabilistas:



- Ejemplo: `ladrón` no es dependiente de `relámpago`. Sin embargo, si se llega a saber que `alarma` es verdadera, entonces bajo las condiciones dadas, `ladrón` y `relámpago` ya no son independientes.

Universidad Veracruzana

Un ejemplo más elaborado

- Las ligas en la red bayesiana sugieren causalidad.
- Ejemplo: El `ladrón` es causa de que el `sensor` se dispare. El `sensor` por su parte, es la causa de que la `alarma` se encienda.
- La estructura de la red permite razonamientos como el siguiente:
- Ejemplo: Si `alarma` es verdadero, entonces un `ladron` pudo entrar en casa, puesto que esa es una de las causas de que la alarma suene.
- Si nos enteramos de que hay una tormenta, la presencia del ladrón debería volverse menos probable.

Universidad Veracruzana

Tipos de razonamiento

- Al sonar la alarma, podemos diagnosticar que la causa posible es la presencia de un ladrón.
- Al enterarnos de la tormenta, podemos predecir que la causa real fue un relámpago.
- En el ejemplo anterior, el razonamiento es de diagnóstico y predictivo al mismo tiempo.

Universidad Veracruzana

Notación

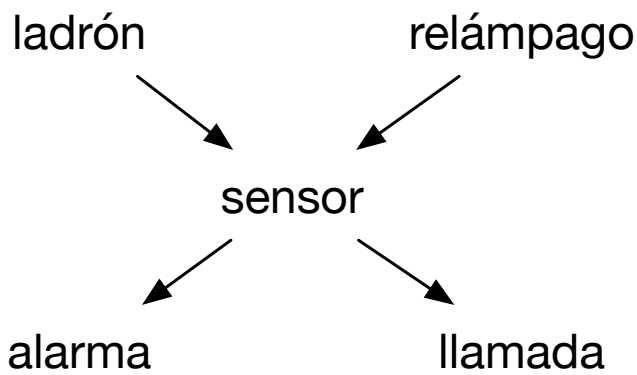
- Un nodo Z es descendiente de X si existe un camino en la red del nodo X al nodo Z.
- Los nodos $Y_1, Y_2, \ldots, Y_n$ son padres de X en la red, si tienen una liga directa hacía X.
- X es independiente de los nodos que no son sus descendientes, dados sus padres (y nada más que ellos).
- De forma que, para computar $P(X)$, basta con tomar en cuenta los descendientes de X y sus padres.
- El efecto de las demás variables en X se acumula en sus padres.

Universidad Veracruzana

Ejemplo

- Supongamos que sabemos que *sensor* es verdadera, y queremos saber $P(alarma)$.
- Dado que ningún nodo en la red es descendientes de `alarma`, y que el valor de *sensor*, único padre de `alarma`, es conocido:

$$p(alarma \mid ladron \wedge relampago \wedge sensor \wedge llamada) = p(alarma \mid sensor)$$



Universidad Veracruzana

Probabilidades de la red

- Para completar la representación de un modelo probabilístico con una red bayesiana, debemos definir algunas probabilidades.
- Para los nodos que no tienen padres (nodos raíz), se especifican sus probabilidades *a priori*.
- Para los demás nodos debemos especificar su probabilidad condicional dado el estado de sus padres: $p(X \mid \text{ padres de } X)$.

Universidad Veracruzana

Nuestra red en Prolog

```
1  %%% Red bayesiana alarma-ladron
2
3  % Estructura de la Red en términos de sucesor (s/2).
4
5  s(ladron, sensor).    % el ladrón tiende a disparar el sensor
6  s(relampago, sensor). % un relámpago fuerte también
7  s(sensor, alarma).
8  s(sensor, llamada).
9
10 % Probabilidades
11
12 p(ladron, 0.001). % Probabilidad a priori
13 p(relampago, 0.02).
14 p(sensor, [ladron, relampago], 0.9). % Probabilidad condicionada
15 p(sensor, [ladron, not relampago], 0.9).
16 p(sensor, [not ladron, relampago], 0.1).
17 p(sensor, [not ladron, not relampago], 0.001).
18 p(alarma, [sensor], 0.95).
19 p(alarma, [not sensor], 0.001).
20 p(llamada, [sensor], 0.9).
21 p(llamada, [not sensor], 0.0).
```

Universidad Veracruzana

Observaciones

- La red tiene dos nodos raíz: *ladron* y *relampago*, con sus probabilidades *a priori* asociadas.
- El nodo *sensor* tiene dos padres, por lo que sus probabilidades condicionadas dado el estado de sus padres forman una tabla de cuatro entradas (2^n, donde n es el número de padres).
- Las variables *alarma* y *llamada* sólo tienen un padre, por lo que sus tablas de probabilidad condicional tienen dos entradas.
- 10 probabilidades, en lugar de $2^5 - 1 = 31$ probabilidades!

Universidad Veracruzana

Cálculo de probabilidades

- Sean X e Y dos proposiciones, entonces:
  - $p(\neg X) = 1 - p(X)$
  - $p(X \wedge Y) = p(X) \times p(Y)$
  - $p(X \vee Y) = p(X) + p(Y) - p(X \wedge Y)$
  - $p(X) = p(X \wedge Y) + (X \wedge \neg Y) = p(Y)p(X \mid Y) + p(\neg Y)p(X \mid \neg Y)$

Universidad Veracruzana

Eventos independientes

- Las proposiciones X es Y se dice independientes si $p(X \mid Y) = p(X)$ y $p(Y \mid X) = p(Y)$.
- Esto es, si Y no afecta mi creencia sobre X y viceversa.
- Si X e Y son independientes, entonces:
  - $p(X \wedge Y) = p(X) \times p(Y)$

Universidad Veracruzana

Eventos disjuntos

- Se dice que las proposiciones X e Y son disjuntas si no pueden ser verdaderas al mismo tiempo, en cuyo caso: $p(X \wedge Y) = 0$ y $p(X \mid Y) = 0$ y $p(Y \mid X) = 0$.

Universidad Veracruzana

Generalización de eventos conjuntos

- Sean $X_1, \ldots, X_n$ proposiciones, entonces:

$$p(X_1 \wedge \cdots \wedge X_n) = p(X_1)p(X_2 \mid X_1)p(X_3 \mid X_1 \wedge X_2) \ldots p(X_n \mid X_1 \wedge \cdots \wedge X_{n-1})$$

- Si todas las variables son independientes, esto se reduce a:

$$p(X_1 \wedge \cdots \wedge X_n) = p(X_1)p(X_2)p(X_3) \ldots p(X_n)$$

Universidad Veracruzana

Teorema de Bayes

- Finalmente, necesitaremos el teorema de Bayes:

$$p(X \mid Y) = p(X)\frac{p(Y \mid X)}{p(Y)}$$

Universidad Veracruzana

Aplicación del Teorema de Bayes

- Consideremos que un ladrón es una causa de que la alarma se encienda, es natural razonar en términos de que proporción de ladrones disparan alarmas, es decir $p(alarma \mid ladron)$.
- Pero cuando oímos la alarma, estamos interesados en saber la probabilidad de su causa, es decir $p(ladron \mid alarma)$. Aquí es donde entra el teorema de Bayes en nuestra ayuda:

$$p(ladron \mid alarma) = p(ladron)\frac{p(alarma \mid ladron)}{p(alarma)}$$

Universidad Veracruzana

Conocimiento previo

- Una variante del teorema de Bayes, toma en cuenta el conocimiento previo B.
- Esto nos permite razonar acerca de la probabilidad de una hipótesis H, dada la evidencia E, bajo el supuesto del conocimiento previo B:

$$p(H \mid E \wedge B) = p(H \mid B) \frac{p(E \mid H \wedge B)}{p(E \mid B)}$$

Universidad Veracruzana

Un intérprete para redes bayesianas

- Dada una red, queremos que este intérprete responda a preguntas del estilo de: ¿Cual es la probabilidad de una proposición dada, asumiendo que otras proposiciones son el caso?
- Ejemplo. Algunas preguntas al intérprete de razonamiento Bayesiano, podrían ser:
  - $p(ladron \mid alarma)$
  - $p(ladron \wedge relampago)$
  - $p(ladron \mid alarma \wedge \neg relampago)$
  - $p(alarma \wedge \neg llamada \mid ladron)$

Universidad Veracruzana

Reglas de cómputo I

1. Probabilidad de una conjunción:

$$p(X_1 \wedge X_2 \mid Cond) = p(X_1 \mid Cond) \times p(X_2 \mid X_1 \wedge Cond)$$

2. Probabilidad de un evento cierto:

$$p(X \mid Y_1 \wedge \cdots \wedge X \wedge \ldots) = 1$$

3. Probabilidad de un evento imposible:

$$p(X \mid Y_1 \wedge \cdots \wedge \neg X \wedge \ldots) = 0$$

4. Probabilidad de una negación:

$$p(\neg X \mid Cond) = 1 - p(X \mid Cond)$$

Universidad Veracruzana

Reglas de cómputo II

5. Si la condición involucra un descendiente de X, usamos el teorema de Bayes. Si Y es un descendiente de X en la red, entonces:

$$p(X \mid Y \wedge Cond) = p(X \mid Cond)\frac{p(Y \mid X \wedge Cond)}{p(Y \mid Cond)}$$

6. Si la condición no involucra descendientes de X, hay dos casos:
   - 6.1 Si X es una causa raíz, entonces $p(X \mid Cond) = p(X)$.
   - 6.2 Si X tiene padres, entonces:

$$p(X \mid Cond) = \sum_{S \in estadosPosibles(Padres)} p(X \mid S)p(S \mid Cond)$$

Universidad Veracruzana

Ejemplo I

- ¿Cual es la probabilidad de un ladrón dado que sonó la alarma?
- Primero, por la regla 5:

$$p(ladron \mid alarma) = p(ladron)\frac{p(alarma \mid ladron)}{p(alarma)}$$

- y por la regla 6:

$$\begin{aligned} p(alarma \mid ladron) \quad = \quad & p(alarma \mid sensor)\ p(sensor \mid ladron) + \\ & p(alarma \mid \neg sensor)p(\neg sensor \mid ladron) \end{aligned}$$

Universidad Veracruzana

Ejemplo II

- y por la misma regla 6:

$$\begin{aligned} p(sensor \mid ladron) = \; & p(sensor \mid ladron \wedge relampago)p(ladron \wedge relampago \mid ladron) + \\ & p(sensor \mid \neg ladron \wedge relampago)p(\neg ladron \wedge relampago \mid ladron) + \\ & p(sensor \mid ladron \wedge \neg relampago)p(ladron \wedge \neg relampago \mid ladron) + \\ & p(sensor \mid \neg ladron \wedge \neg relampago)p(\neg ladron \wedge \neg relampago \mid ladron) \end{aligned}$$

- Aplicando las reglas 1,2,3 y 4 como corresponde, esto se reduce a:

$$\begin{aligned} p(sensor \mid ladron) &= 0{,}9 \times 0{,}02 + 0 + 0{,}9 \times 0{,}98 + 0 = 0{,}9 \\ p(alarma \mid ladron) &= 0{,}95 \times 0{,}9 + 0{,}001 \times (1 - 0{,}9) = 0{,}8551 \end{aligned}$$

- Usando las reglas 1,4 y 6 obtenemos:

$$p(alarma) = 0{,}00467929$$

Universidad Veracruzana

Ejemplo III

- Finalmente

$$p(ladron \mid alarma) = 0{,}001 \times 0{,}8551/0{,}00467929 = 0{,}182741$$

Universidad Veracruzana

En Prolog I

```
1  % Motor de inferencia para Red Bayesiana
2
18
19 :-  op(900, fy, not). % not definido como operador prefijo
20
21 prob([X|Xs], Cond, P)  :-  !,   % Prob de la conjunción
22   prob(X, Cond, Px),
23   prob(Xs, [X|Cond], PRest),
24   P is Px * PRest.
25
26 prob([], _, 1)  :-  !.            % Conjunción vacía
27
28 prob(X, Cond, 1)  :-
29   member(X, Cond), !.            % Cond implica X
30
31 prob(X, Cond, 0)  :-
32   member(not X, Cond), !.        % Cond implica X es falsa
33
34 prob(not X, Cond, P)  :-  !,     % Negación
35   prob(X, Cond, P0),
36   P is 1 - P0.
```

Universidad Veracruzana

En Prolog II

```
37
38  % Usa la regla de Bayes si Cond0 incluye un descendiente de X
39
40  prob(X,Cond0,P)  :-
41    select(Y,Cond0,Cond),
42    pred(X,Y), !,            % Y es un descendiente de X
43    prob(X,Cond,Px),
44    prob(Y,[X|Cond], PyDadoX),
45    prob(Y,Cond,Py),
46    P is Px * PyDadoX / Py.          % Asumiendo Py > 0
47
48  % Casos donde Cond no involucra a un descendiente
49
50  prob(X, _, P)  :-
51    p(X, P), !.                    % X una causa raíz
52
53  prob(X, Cond, P)  :-  !,
54    findall((Condi,Pi), p(X,Condi,Pi), CPlist),  % Conds padres
55    suma_probs(CPlist, Cond, P).
56
57  suma_probs([], _, 0).
```

Universidad Veracruzana

En Prolog III

```
58
59  suma_probs([(Cond1,P1)|CondsProbs], Cond, P)  :-
60    prob(Cond1, Cond, PC1),
61    suma_probs(CondsProbs, Cond, PResto),
62    P is P1 * PC1 + PResto.
63
64  % pred(X, Y). El nodo X es predecesor del nodo Y en la Red Bayesiana.
65
66  pred(X, not Y)  :- !,          % Y negada
67    pred(X, Y).
68
69  pred(X, Y)  :-
70    s(X, Y).
71
72  pred(X, Z)  :-
73    s(X, Y),
74    pred(Y, Z).
```

Universidad Veracruzana

Consultas

```
?- prob(ladron,[alarma],P).
P = 0.182741321476588.

?- prob(alarma,[],P).
P = 0.00467929198.

?- prob(ladron,[llamada],P).
P = 0.232137053716514 22.

?- prob(ladron,[llamada,relampago],P).
P = 0.00892857142857142 8.

?- prob(ladron,[llamada,not relampago],P).
P = 0.47393364928909953.
```

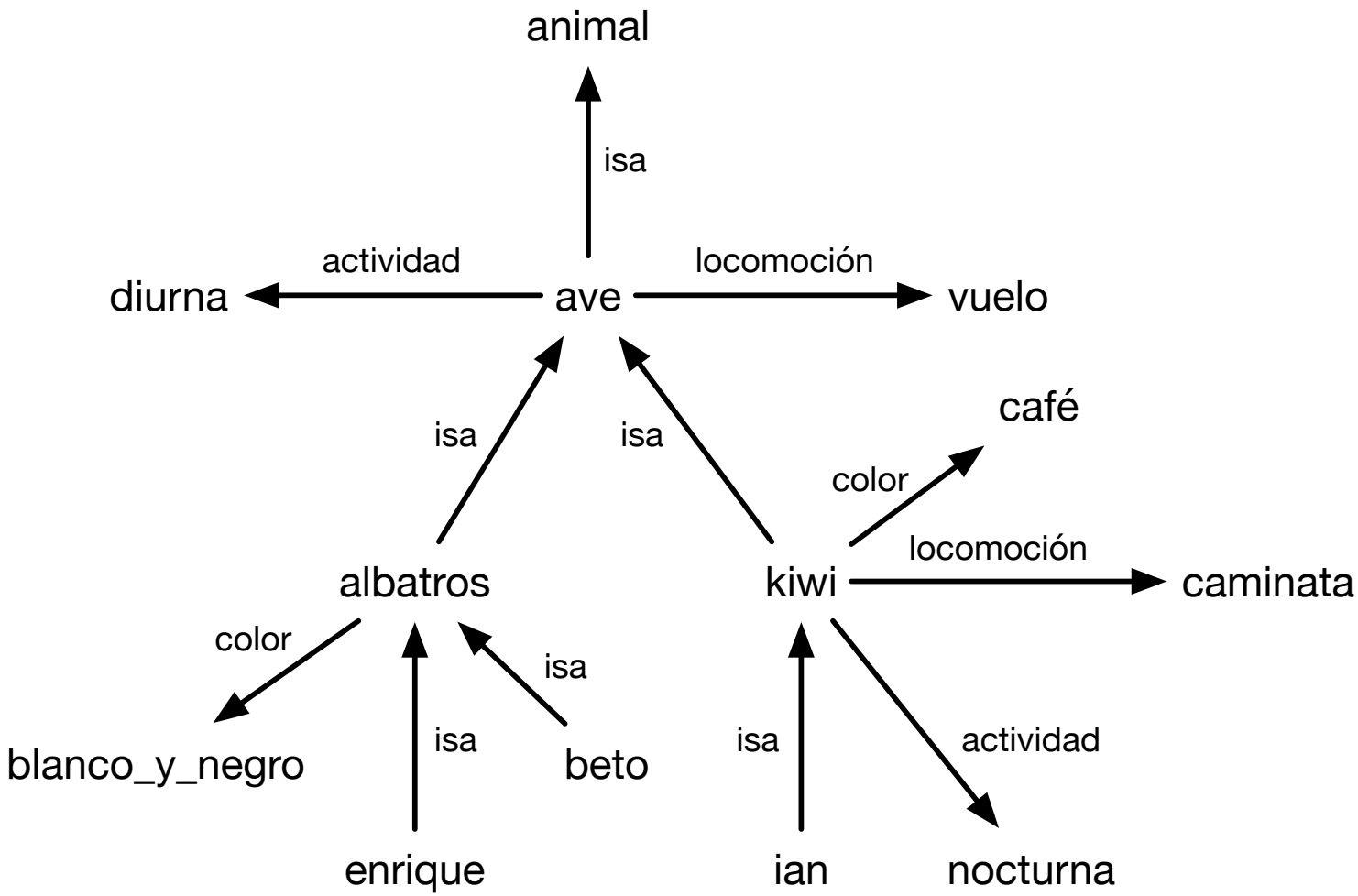
Universidad Veracruzana

Redes semánticas y Marcos

- Están diseñados para representar, de manera estructurada, grandes cantidades de hechos.
- Este conjunto de hechos es normalmente compactado: Se obvian algunos hechos, cuando estos pueden ser inferidos.
- Utilizan la herencia, de manera similar a como se utiliza en los lenguajes de programación orientados a objetos.

Universidad Veracruzana

Red semántica



Universidad Veracruzana

Relaciones y herencia

- La relación especial `isa` denota la relación es un. La red del ejemplo representa los siguientes hechos:
  - Un ave es un tipo de animal.
  - El vuelo es el medio de locomoción de las aves.
  - Un albatros es un ave.
  - Enrique y Beto son albatros.
  - Un kiwi es un ave.
  - La caminata es el medio de locomoción de un kiwi.
  - etc.

Universidad Veracruzana

Red semántica en Prolog

```
1  % Red semántica
2
3  isa(ave,animal).
4  isa(albatros,ave).
5  isa(kiwi,ave).
6  isa(enrique,albatros).
7  isa(beto,albatros).
8  isa(ian,kiwi).
9
10 actividad(ave,diurna).
11 actividad(kiwi,nocturna).
12
13 color(albatros,blanco_y_negro).
14 color(kiwi,cafe).
15
16 locomocion(ave,vuelo).
17 locomocion(kiwi,caminata).
```

Universidad Veracruzana

Inferencias sobre red semántica I

```
1  % Interprete Redes Semánticas
2
3  fact(Hecho) :-
4      Hecho,!.
5
6  fact(Hecho) :-
7      Hecho =.. [Rel,Arg1,Arg2],
8      isa(Arg1,SuperClaseArg1),
9      SuperHecho =.. [Rel,SuperClaseArg1,Arg2],
10     fact(SuperHecho).
11
12
```

Universidad Veracruzana

Consulta a red semántica

```
?- [seRedSemantica].
true.
?- fact(locomocion(ian,Loc)).
Loc = caminata.
?- fact(locomocion(enrique,Loc)).
Loc = vuelo.
```

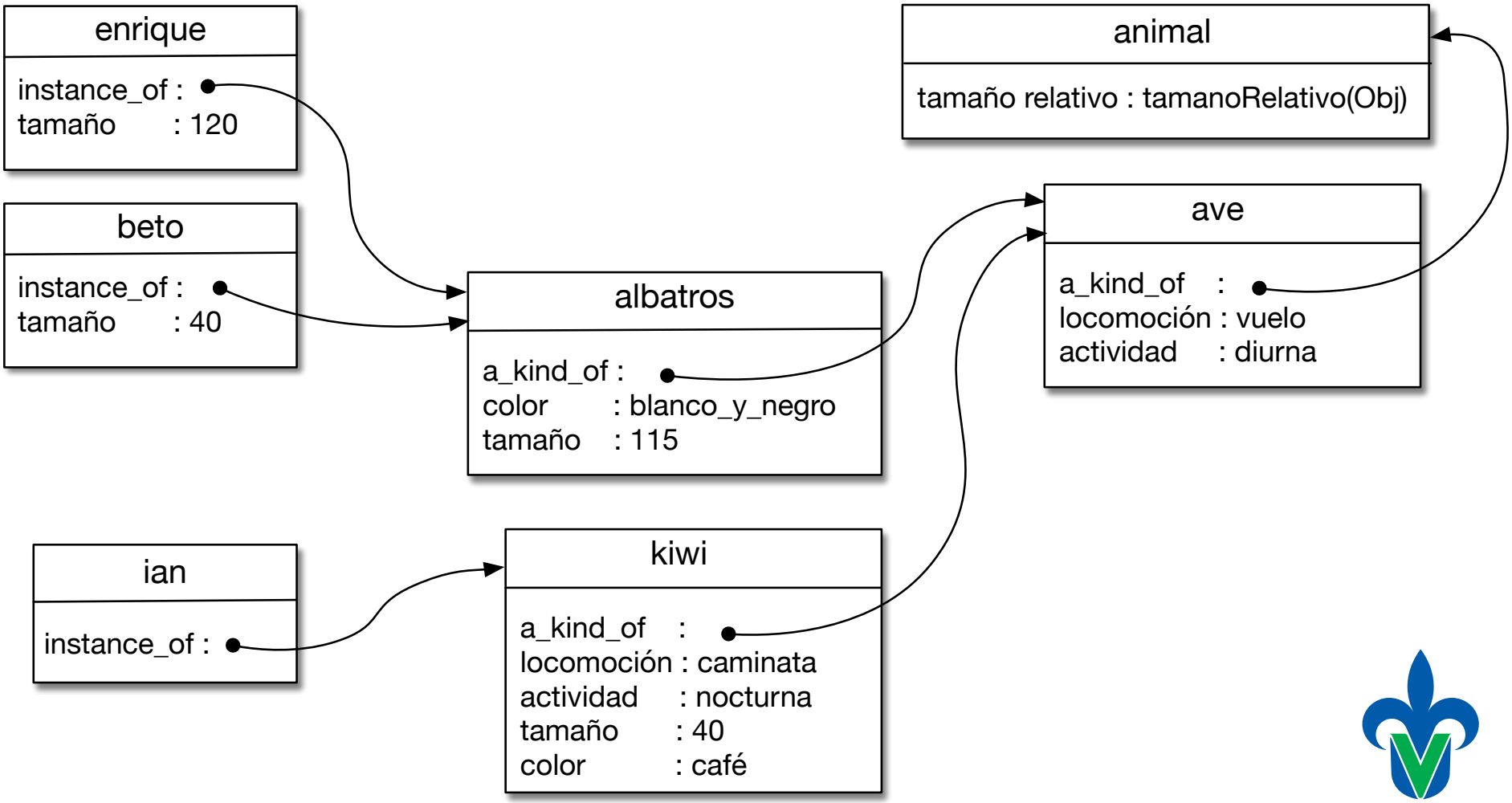
Universidad Veracruzana

Marcos

- Predecesor de la programación orientada a objetos.
- Un marco (*frame*) es una estructura de datos, cuyos componentes se llaman ranuras (*slots*).
- Las ranuras pueden guardar información de diferentes tipos:
  - Valores.
  - Referencias a otros marcos.
  - Procedimientos para computar valores.
- Tambien usan herencia

Universidad Veracruzana

Representación basada en marcos.



Universidad Veracruzana

Marcos en Prolog I

```
1  % Base de conocimiento basada en Marcos
2  % Los hechos se representan como marco(ranura,valor).
3  % El valor puede ser un procedimiento para su cálculo.
4
5  % Marco ave: un ave prototípica
6
7  ave(a_kind_of,animal).
8  ave(locomocion,vuelo).
9  ave(actividad,diurna).
10
11 % Marco albatros: un ave prototípica con hechos extra:
12 % es blanco y negro; y mide 115cm
13
14 albatros(a_kind_of,ave).
15 albatros(color,blanco_y_negro).
16 albatros(tamano,115).
17
18 % Marco kiwi: un ave prototípica con hechos extra:
19 % camina, es nocturno, café y mide unos 40cm
20
21 kiwi(a_kind_of,ave).
22 kiwi(locomocion,caminata).
```

Universidad Veracruzana

Marcos en Prolog II

```
23  kiwi(actividad,nocturna).
24  kiwi(tamano,40).
25  kiwi(color,cafe).
26
27  % Marco enrique: es un albatros
28
29  enrique(instance_of,albatros).
30  enrique(tamano,120).
31
32  % Marco beto: es un albatros bebé
33
34  beto(instance_of,albatros).
35  beto(tamano,40).
36
37  % Marco ian: es un kiwi.
38
39  ian(instance_of,kiwi).
40
41  % Marco animal: su tamaño relativo se calcula ejecutando un
42  % procedimiento
43
```

Universidad Veracruzana

Marcos en Prolog III

```
44  animal(tamanoRelativo,
45         execute(tamanoRelativo(Obj,Val), Obj, Val)).
46
47  % tamanoRelativo(Obj,TamRel): El tamaño relativo TamRel de Obj
48
49  tamanoRelativo(Obj,TamRel) :-
50      value(Obj,tamano,TamObj),
51      value(Obj,instance_of,ClaseObj),
52      value(ClaseObj,tamano,TamClase),
53      TamRel is (TamObj/TamClase) * 100.
```

Universidad Veracruzana

Inferencia con marcos en Prolog I

```
1  % Motor de inferencia para Marcos
2
3  value(Frame,Slot,Value) :-
4      Query =.. [Frame,Slot,Value],
5      Query, !. % valor encontrado directamente.
6
7  value(Frame,Slot,Value) :-
8      parent(Frame,ParentFrame), % Un marco más general
9      value(ParentFrame,Slot,Value).
10
11
12 parent(Frame,ParentFrame) :-
13     (Query =.. [Frame, a_kind_of, ParentFrame]
14     ;
15     Query =.. [Frame, instance_of, ParentFrame]
16     ),
17     Query.
```

Universidad Veracruzana

Consultas a los marcos

```
?- value(enrique,actividad,Act).  Act = diurna

?- value(kiwi,actividad,Act).  Act = nocturna.
```

Universidad Veracruzana

Inferencia con marcos y funciones en Prolog I

```
% Motor inferencial para marcos V2: Incluye funciones

value(Frame,Slot,Value) :-
    value(Frame, Frame, Slot, Value).

value(Frame, SuperFrame, Slot, Value) :-
    Query =.. [SuperFrame, Slot, ValAux],
    Query,
    process(ValAux, Frame, Value), !.

value(Frame, SuperFrame, Slot, Value) :-
    parent(SuperFrame, ParentSuperFrame),
    value(Frame, ParentSuperFrame, Slot, Value).

parent(Frame,ParentFrame) :-
    (Query =.. [Frame, a_kind_of, ParentFrame]
    ;
    Query =.. [Frame, instance_of, ParentFrame]
    ),
    Query.

process(execute(Proc, Frame, Value), Frame, Value) :- !,
```

Universidad Veracruzana

Inferencia con marcos y funciones en Prolog II

```
23        Proc.
24
25  process(Value,_,Value).   % un valor, no un procedimiento.
```

Universidad Veracruzana

Consultas

```
?- value(enrique,tamanoRelativo,Tam).
Tam = 104.34782608695652
?- value(beto,tamanoRelativo,Tam).
Tam = 34.78260869565217
?- value(ian,tamanoRelativo,Tam).
Tam = 100
?- value(beto,color,C).
C = blanco_y_negro.
```

Universidad Veracruzana

Interfaz

- La base de conocimientos basada en marcos y su motor de inferencia, incluyendo funciones, se puede llamar con el *script* `seMarcos.pl`:

```
1  % Sistema Experto basado en marcos.
2
3  % :- [inferenciaMarcos].
4  :- [inferenciaMarcosFunc].
5  :- [kbMarcos].
```

Universidad Veracruzana

Referencias I

[1] VE Barker y O'Connor. "Expert systems for configuration at Digital: XCON and beyond". En: *Communications of the ACM* 32.3 (1989), págs. 298-318.

[2] I Bratko. *Prolog programming for Artificial Intelligence*. Fourth. Essex, England: Pearson, 2012.

[3] BG Buchanan y EH Shortliffe. *Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project*. The Addison-Wesley Series in Artificial Intelligence. Reading, MA, USA: Addison-Wesley, 1984.

[4] PE Hart, RO Duda y MT Einaudi. "PROSPECTOR– A Computer-Based Consultation System for Mineral Exploration". En: *Mathematical Geology* 10.5 (1978), págs. 589-610.

[5] RK Lindsay et al. *Applications of Artificial Intelligence for Organic Chemestry: The DENDRAL Project*. McGraw-Hill advanced computer science series. New York, NY, USA: McGraw-Hill Book Company, 1980.

[6] RK Lindsay et al. "DENDRAL: a case study of first expert system for scientific hypothesis formation.". En: *Artificial Intelligence* 61.2 (1993), págs. 209-261.

[7] J McDermott. "R1: A rule-based configurer of computer systems". En: *Artificial intelligence* 19.1 (1982), págs. 39-88.

Universidad Veracruzana

Referencias II

[8] EH Shortliffe et al. "An Artificial Intelligence program to advise physicians regarding antimicrobial therapy". En: *Computers and Biomedica Research* 6.6 (1973), págs. 544-560.

[9] EH Shortliffe et al. "Computer-Based Consultations in Clinical Therapeutics: Explanation and Rule Acquisition Capatilities of the MYCIN System". En: *Computers and Biomedica Research* 8.4 (1975), págs. 303-320.

Universidad Veracruzana