Técnicas de Inteligencia Artificial

Tema 2

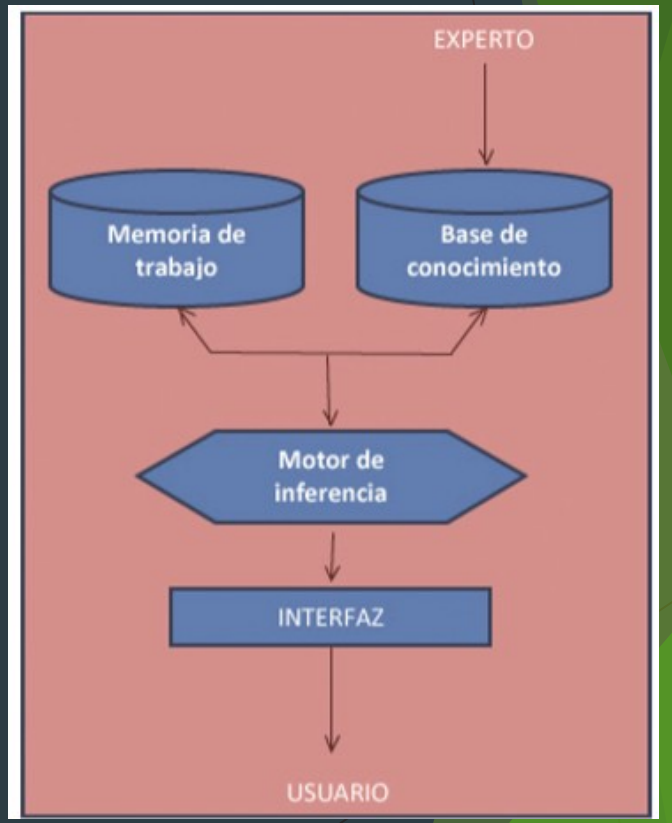
Razonamiento monotónico y no monotónico

Sistemas expertos

- Un experto es un humano especializado y con grandes conocimientos en una materia.
- Objetivos: capturar el conocimiento humano en una computadora y replicar el comportamiento del experto.
  - Codificar el conocimiento humano experto (Representación del conocimiento).
  - Aplicar el conocimiento para resolver problemas (Razonamiento)
  - Explicar la resolución del problema (Introspección).
- Capacidades:
  - Resolver problemas difíciles y apoyar la toma de decisiones con razonamiento simbólico.
  - Soportar incertidumbre.
  - Ofrecer interfaz natural, eficaz y cómoda para interactuar con el conocimiento.
  - Incorporar posibilidad de aprendizaje.
  - Ofrecer soluciones alternativas y justificar los razonamientos.

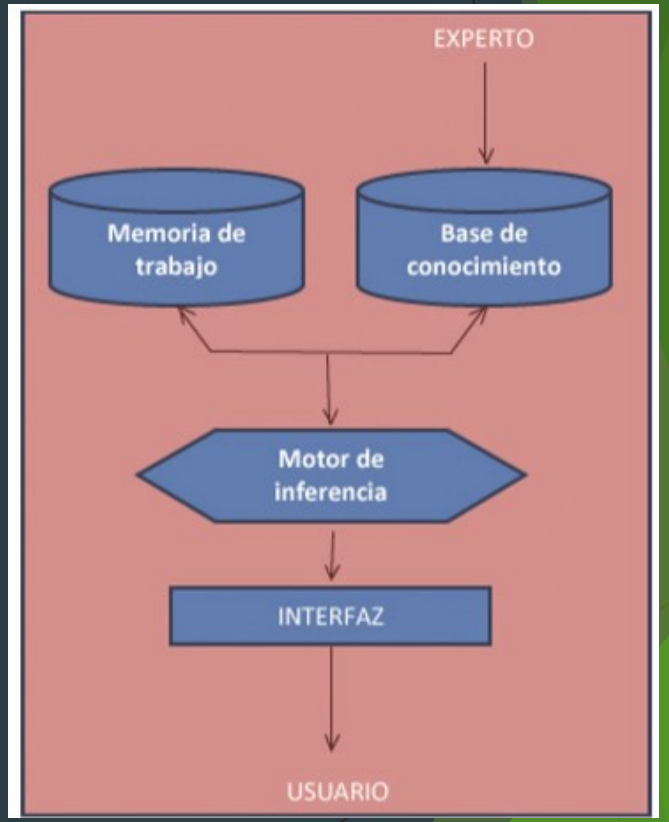
Sistemas expertos

- **Arquitectura del sistema experto**
  - **Base de conocimiento**.
    - Conocimiento altamente especializado.
    - Proporcionado por el experto humano.
    - Hechos, reglas, conceptos y relaciones.
    - Conocimiento sobre el dominio.
    - Típicamente reglas SI/ENTONCES.
  - **Base de hechos.**
    - Memoria de trabajo.
    - Hechos descubiertos durante una consulta.
    - Durante una consulta el usuario introduce información del problema actual en la base de hechos.



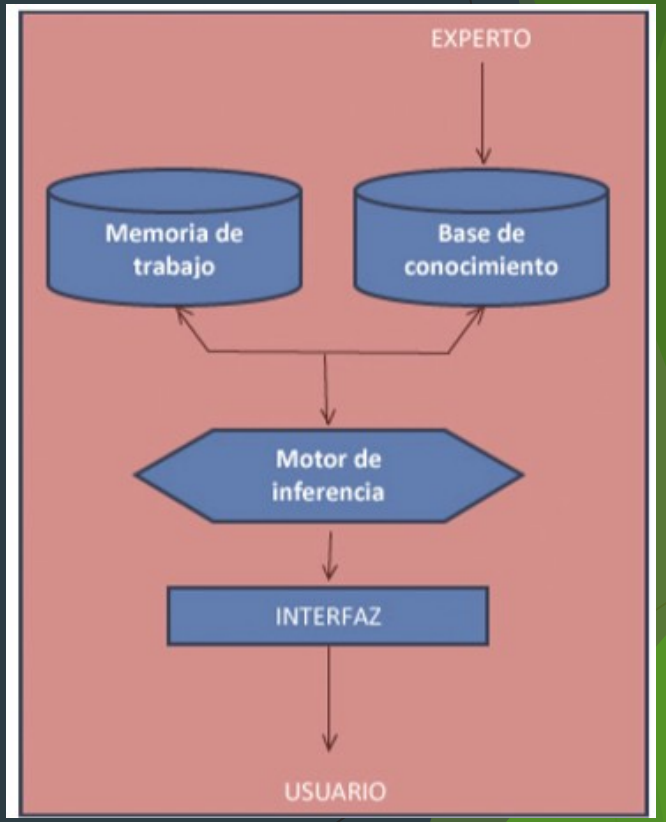
Sistemas expertos

- **Arquitectura del sistema experto**
  - **Motor de inferencia**.
    - Realiza el razonamiento a partir de los hechos y el conocimiento en la base de conocimiento: las reglas.
    - Deduce nuevos hechos.
    - Obtiene conclusiones.
    - Tipos:
      - Determinista. Las conclusiones obtenidas se asumen como verdad. Basado en hechos y reglas.
      - Probabilístico. Las conclusiones contienen incertidumbre. Basados en probabilidad.
    - Modos de derivación de soluciones:
      - Forward Chaining. Comienza en los hechos conocidos y reglas y aplica reglas de inferencia para obtener las conclusiones.
      - Backward Chaining: Comienza a partir del objetivo y realiza un proceso de retroceso para probar los hechos conocidos.

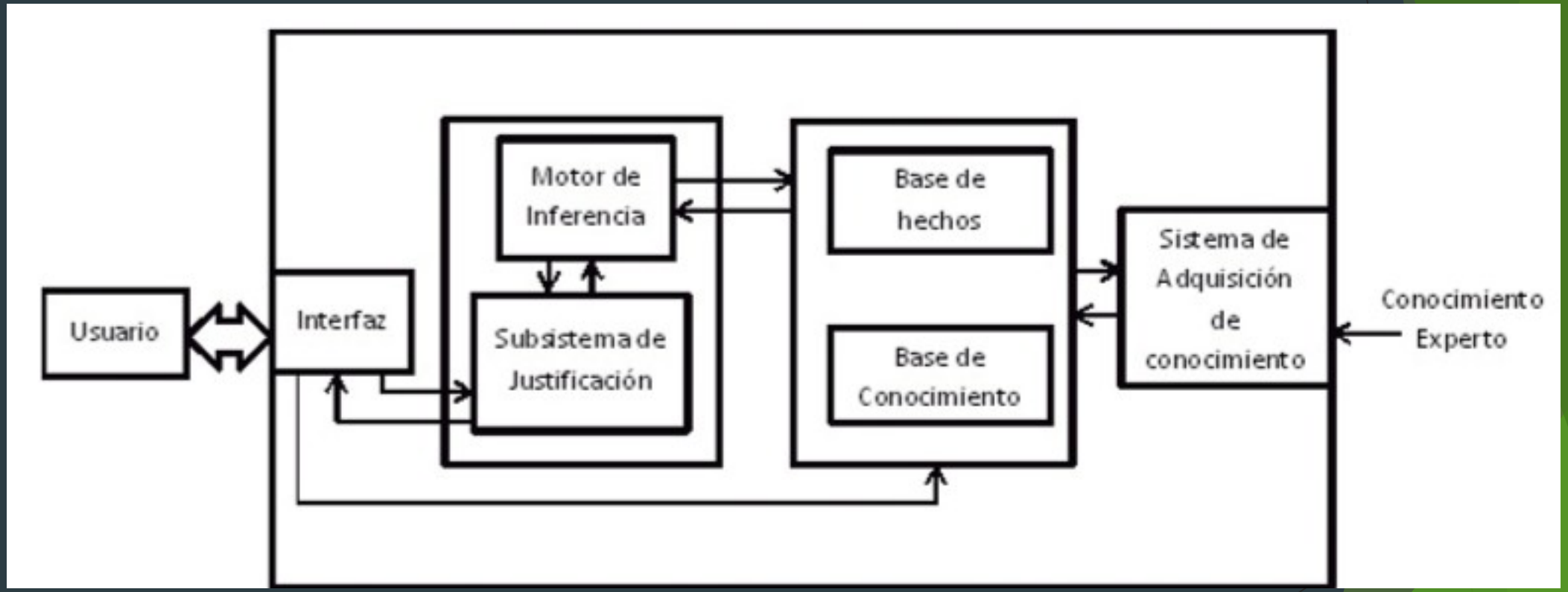


Sistemas expertos

- **Arquitectura del sistema experto**
  - **Subsistema de explicación.**
    - Explica las conclusiones.
    - Proporciona información de por qué plantea nuevas preguntas al usuario y cómo ha llegado a sus conclusiones.
  - **Interfaz de usuario.**
    - Interacción en lenguaje natural.
    - Habilidad para hacer preguntas.
    - Presenta resultados gráficamente.
  - **Subsistema de adquisición del conocimiento.**
    - Acumulación, transferencia y transformación de la experiencia para resolver problemas de una fuente de conocimiento.
    - Construir o expandir la base de conocimiento.
    - Requiere un ingeniero en conocimiento que interactué con uno o más expertos humanos para construir la base de conocimiento.



Sistemas expertos



Sistemas expertos

- **Tipos**.
  - *Control*. Gobiernan el comportamiento de un sistema.
  - *Diseño*. Configuran objetos bajo un conjunto de restricciones.
  - *Diagnóstico*. Deducen fallos del sistema desde información observable.
  - *Instrucción*. Guían la educación de un estudiante en un tema dado.
  - *Interpretación*. Producen una descripción de una situación a partir de la información disponible.
  - *Monitorización*. Comparan información del comportamiento de un sistema con estados “cruciales” en su operación.
  - *Planificación*. Diseñan un plan de acciones para realizar un objetivo dado.
  - *Predicción*. Deducen consecuencias probables desde una situación dada y un modelo del problema.
  - *Selección*. Identifican la mejor elección de una lista de posibilidad.
  - *Simulación*. Modelan un proceso o sistema para permitir estudios operaciones bajo diversas condiciones.

Sistemas expertos

- **Lenguajes de programación**. Lenguajes de programación simbólicos de alto nivel.
  - LISt Programming (LISP)
  - PROgrammation en LOGique (PROLOG).
    - Ejemplo de sistema experto para navegación semántica en PROLOG:

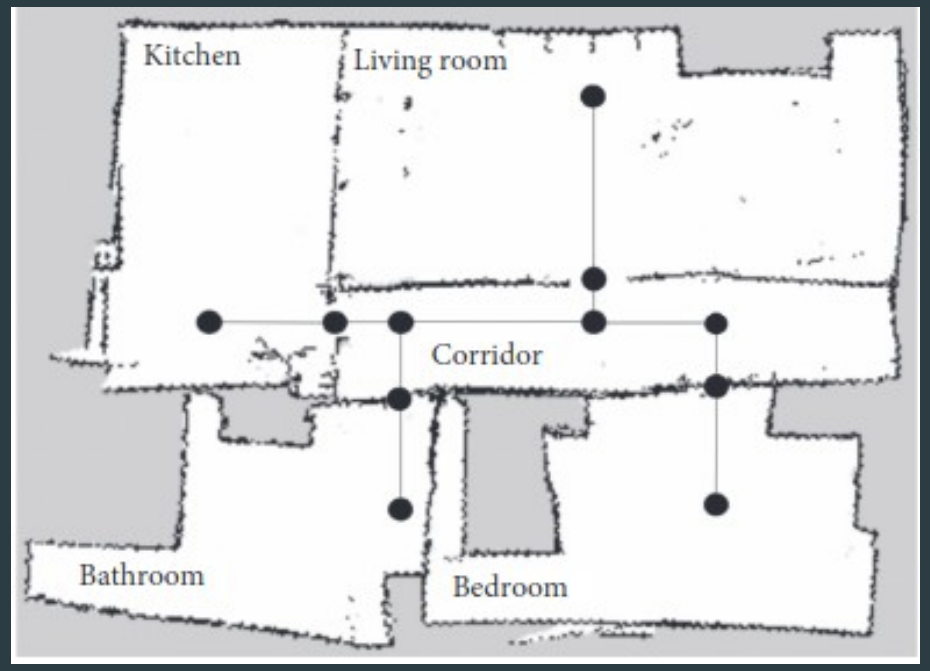


Table 1: Transition matrix between node types.

From/to	Room	Door	Cross	Interior
Room	cd	cd	ND	fc
Door	cd	fc	fc	cd
Cross	ND	fc	fc	ND
Interior	fc	cd	ND	fc

```
(1) %Type definitions
(2) room (kitchen).
(3) room (corridor).
(4) room (bathroom).
(5) room (living_room).
(6) room (bedroom).
(7) door (door_one).
(8) door (door_two).
(9) door (door_three).
(10) door (door_four).
(11) interior (interior_corridor_one).
(12) interior (interior_corridor_two).
(13) interior (interior_corridor_three).
(14) %Types without examples
(15) :-dynamic
(16)     cross/1.
(17) %Dynamic predicate to indicate closed doors
(18) :-dynamic
(19)     closed/1.
```

Sourcecode 3: Definition of node types facts.

Sistemas expertos

```
(1) %Directions
(2) dir (north).
(3) dir (south).
(4) dir (east).
(5) dir (west).
(6) %Reversibility of orientations
(7) revDir (east, west).
(8) revDir (north, south).
(9) %Revert orientations
(10) isRevDir (X, Y):- revDir (X, Y).
(11) isRevDir (X, Y):- revDir (Y, X).
(12) % Look for direct connections
(13) hasConnection (X, X, none, 0).
(14) hasConnection (X, Y, Direction, Cost):- dir (Direction), connection (X, Y, Direction, Cost).
(15) hasConnection (X, Y, Direction, Cost):- dir (Direction), isRevDir (Direction, Reversed),
(16)     connection (Y, X, Reversed, Cost).
```

SOURCECODE 1: Definition of connection rules.

```
(1) %Action definition
(2) action (X, X, none):- isPlace (X).
(3) action (X, Y, cd):- room (X), room (Y), X\ = Y.
(4) action (X, Y, fc):- cross (X), cross (Y), X\ = Y.
(5) action (X, Y, fc):- door (X), door (Y), X\ = Y.
(6) action (X, Y, fc):- interior (X), interior (Y), X\ = Y.
(7) action (X, Y, cd):- room (X), door (Y), not (closed (Y)), X\ = Y.
(8) action (X, Y, fc):- room (X), interior (Y), X\ = Y.
(9) action (X, Y, fc):- cross (X), door (Y), X\ = Y.
(10) action (X, Y, cd):- door (X), not (closed (X)), interior (Y), X\ = Y.
(11) %Know if X is an existing place
(12) isPlace (X):- room (X).
(13) isPlace (X):- door (X).
(14) isPlace (X):- interior (X).
(15) isPlace (X):- cross (X).
(16) %Action rules than ensures reversibility
(17) isAction (X, X, Action):- action (X, X, Action).
(18) isAction (X, Y, Action):- action (X, Y, Action), X\ = Y, !.
(19) isAction (X, Y, Action):- action (Y, X, Action), X\ = Y, !.
```

SOURCECODE 2: Definition of action rules.

```
?- goToFrom (kitchen, bedroom, Path, Actions, Directions, Cost).
Path = [kitchen, door_one, interior_corridor_one, corridor, interior_corridor_two, interior_corridor_three, door_four, bedroom],
Actions = [cd, cd, fc, fc, fc, cd, cd],
Directions = [east, east, east, east, east, south, south],
Cost = 700.
```

EXECUTION RESULT 1: Execution result from kitchen to bedroom.

Sistemas expertos

- Sistema experto vs experto humano

Factor	Experto humano	Sistema experto
Disponibilidad de tiempo	Día laborable	Siempre
Geografía	Local	Cualquier lugar
Seguridad	Irremplazable	Reemplazable
Caducidad	Sí	No
Ejecución	Variable	Constante
Velocidad	Variable	Constante
Coste	Alto	Abordable

Sistemas expertos

- **Ventajas**.
  - Reutilización del motor de inferencia para distintos problemas.
  - Facilita el proceso de aprendizaje de nuevo conocimiento experto (y se puede automatizar).
  - Facilita el manejo de conocimiento incompleto.
  - Se centra en un problema específico apartando problemas referentes a otros conocimientos.
  - Permanencia, transportabilidad, coherencia, coste y estabilidad.
  - Sus decisiones no están afectadas por emociones humanas.
  - Sin limitaciones de memoria.

Sistemas expertos

- **Desventajas**.
  - Resolución y búsqueda menos eficientes al separar el conocimiento sobre el dominio y sobre el proceso.
  - No capturan algunas buenas cualidades de los expertos humanos como creatividad, sentido común, adaptabilidad o amplitud de miras.
  - Comprobación costosa de inconsistencias.
  - La respuesta del sistema puede ser incorrecta si la información almacenada no es verdadera.
  - No puede aprender por sí mismo y requiere de actualizaciones manuales.

Sistemas expertos

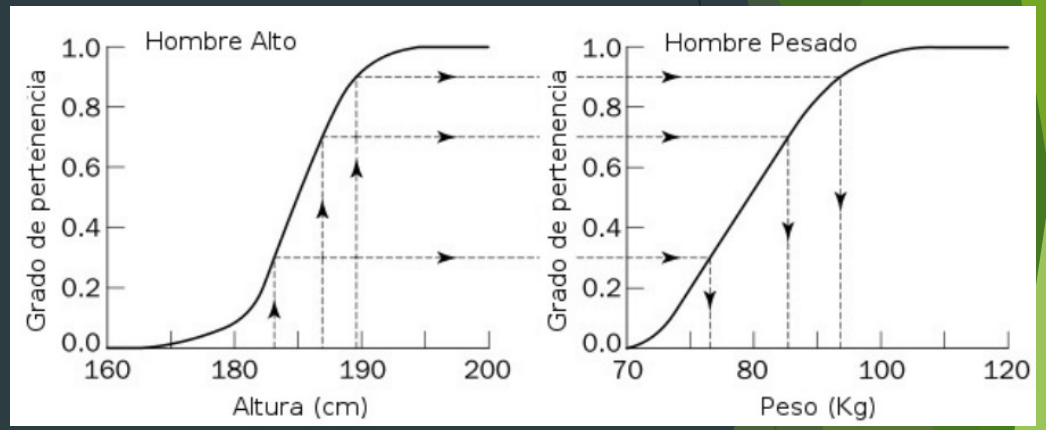
- **Ámbitos de uso**.
  - Servicios financieros. Tomar decisiones sobre gestión de activos, actúan como roboasesores y hacen predicciones sobre el comportamiento de diversos mercados y otros indicadores financieros.
  - Ingeniería mecánica. Solucionar problemas de maquinaria electromecánica compleja.
  - Telecomunicaciones. Tomar decisiones sobre las tecnologías de red utilizadas y el mantenimiento de las redes existentes.
  - Sanidad. Ayudar en los diagnósticos médicos.
  - Agricultura. Pronosticar los daños en las cosechas.
  - Atención al cliente. Ayudar a programar pedidos, encauzar las solicitudes de los clientes y resolver problemas.
  - Transporte. Control de semáforos, diseño de autopistas, programación y mantenimiento de autobuses y trenes, patrones de vuelo de la aviación y control del tráfico aéreo.
  - Derecho. Prestar servicios jurídicos, realizar evaluaciones de casos civiles y evaluar la responsabilidad de los productos.

Sistemas expertos

- **Ejemplos**.
  - CaDet (Cancer Decision Support Tool). Usado para identificar el cáncer en etapas tempranas.
  - DENDRAL. Ayuda a los químicos a identificar moléculas orgánicas desconocidas.
  - Dxplain. Sistema de soporte clínico que diagnostica varias enfermedades.
  - MYCIN. Identifica bacterias como la bacteriemia o la meningitis, y recomienda antibióticos y dosis.
  - PXDES. Determina el tipo y la severidad del cáncer de pulmón.
  - R1/XCON. Automáticamente selecciona y ordena componentes de ordenador según las especificiaciones del cliente.

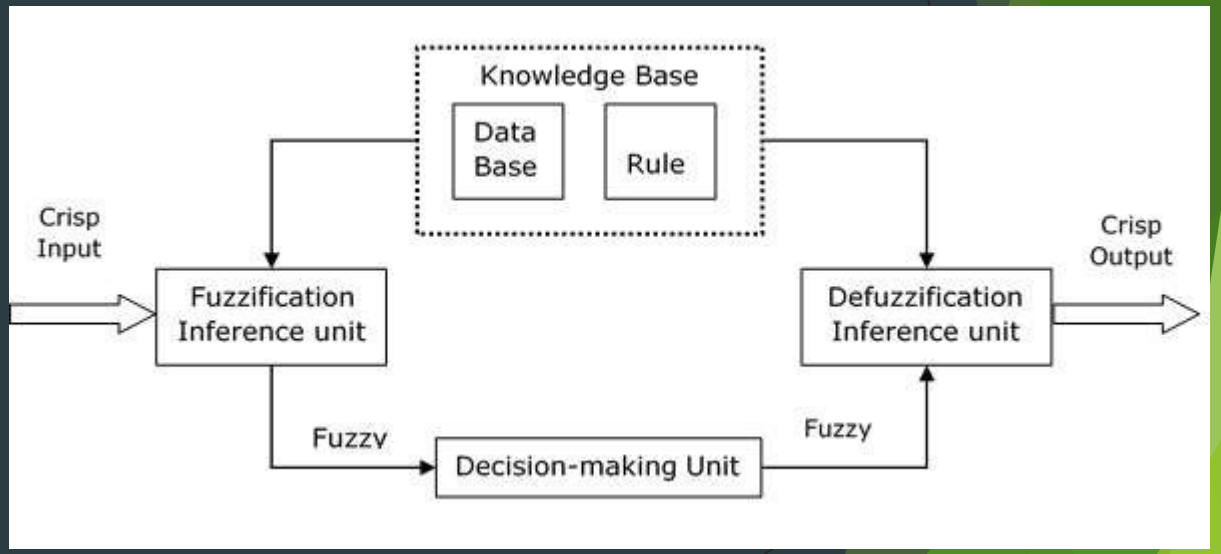
Sistemas expertos difusos

- Utilizan la lógica difusa para realizar un razonamiento aproximado.
  - Ejemplos:
    - “El coche es pequeño”.
    - “Tania es muy alta”.
    - “El movimiento es lento”.
- Utilizan reglas difusas.
  - IF <proposición difusa> THEN <proposición difusa>
  - Ejemplo: IF altura IS alto THEN peso IS pesado.
  - El antecedente y el consecuente de una regla pueden tener múltiples partes.
    - IF a IS x AND b IS y THEN c IS z.



Sistemas expertos difusos

- **Inferencia difusa** (inferencia de Mamdani).
  - Obtener un valor de salida para un valor de entrada usando la teoría de conjuntos difusos.
  - Etapas.
    1) Fuzificación de las variables de entrada.
    2) Evaluación de las reglas.
    3) Agregación de las salidas de las reglas.
    4) Defuzificación.

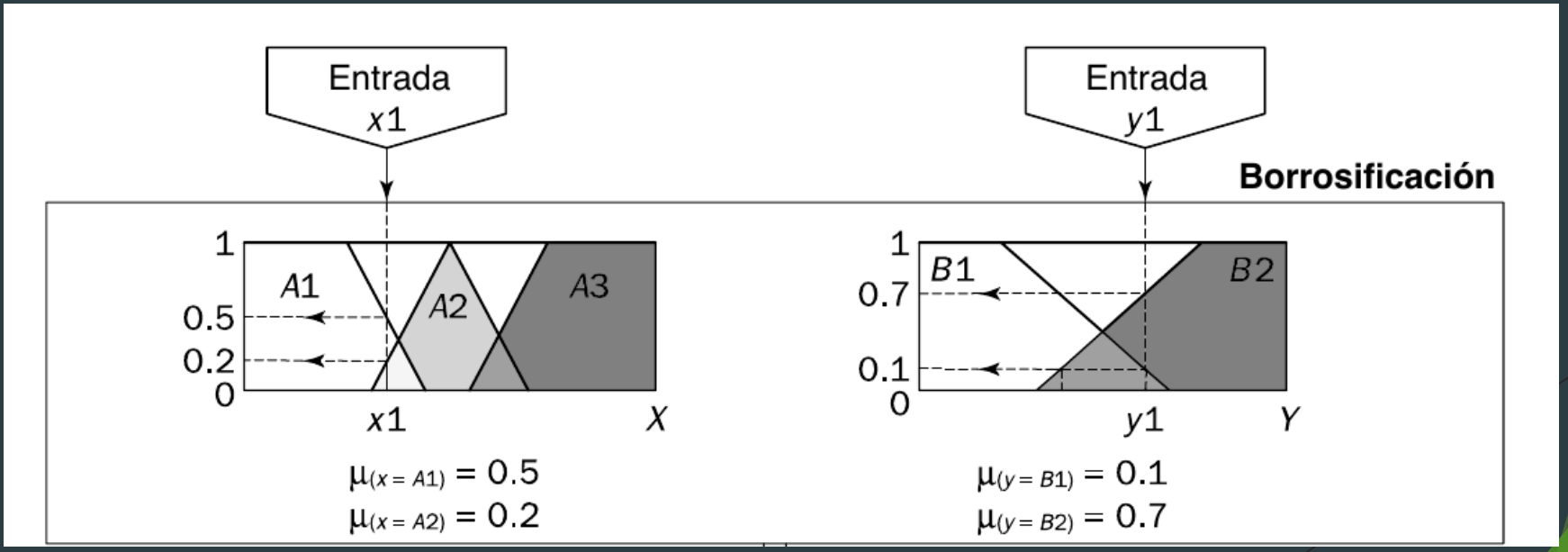


Sistemas expertos difusos

- Ejemplo de inferencia difusa.
  - **Variables lingüísticas**.
    - x (financiación-del-proyecto)
    - y (plantilla-del-proyecto)
    - z (riesgo)
  - **Conjuntos difusos** definidos sobre los dominios de las variables:
    - X: A1, A2, A3 (inadecuado, marginal, adecuado)
    - Y: B1, B2 (pequeña, grande)
    - Z: C1, C2, C3 (bajo, normal, alto)

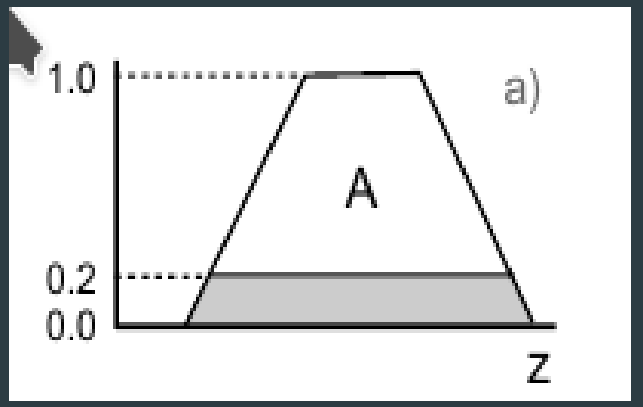
Sistemas expertos difusos

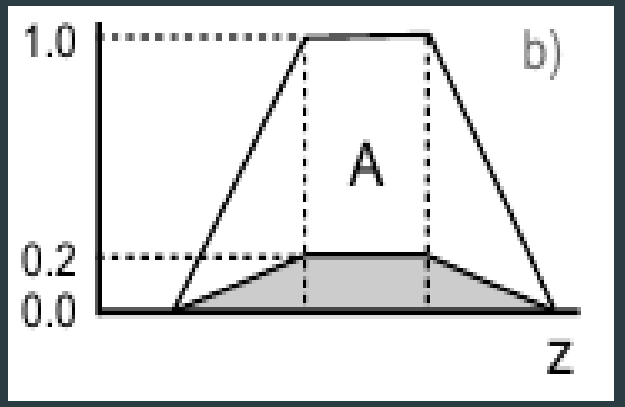
- Ejemplo de inferencia difusa.
  - **Etapa 1. Fuzificación (Borrosificación)**.
    - Tomar los valores de las entradas y determinar el grado de pertenencia a los conjuntos difusos.
    - Vamos a suponer x=35%, y=60%



Sistemas expertos difusos

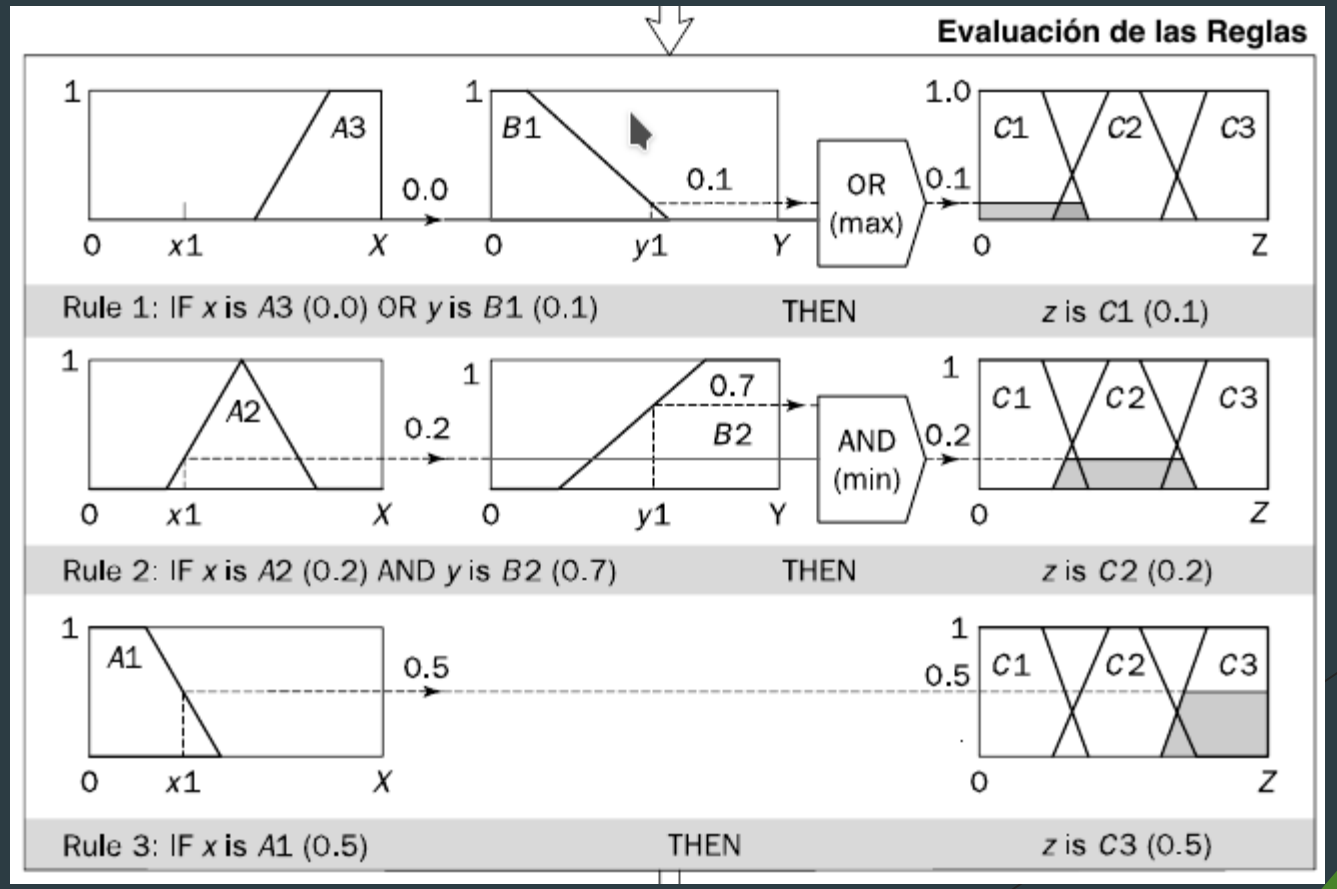
- Ejemplo de inferencia difusa.
  - **Etapa 2. Evaluación de las reglas**.
    - Los valores fuzificados se aplican a los antecedentes de TODAS las reglas de producción.
    - En caso de antecedentes compuestos por conectivas, se aplican las funciones t-conorma y t-norma.
    - El resultado de la evaluación del antecedente se aplica al consecuente, aplicando un recorte o escalado según el valor de verdad del antecedente.





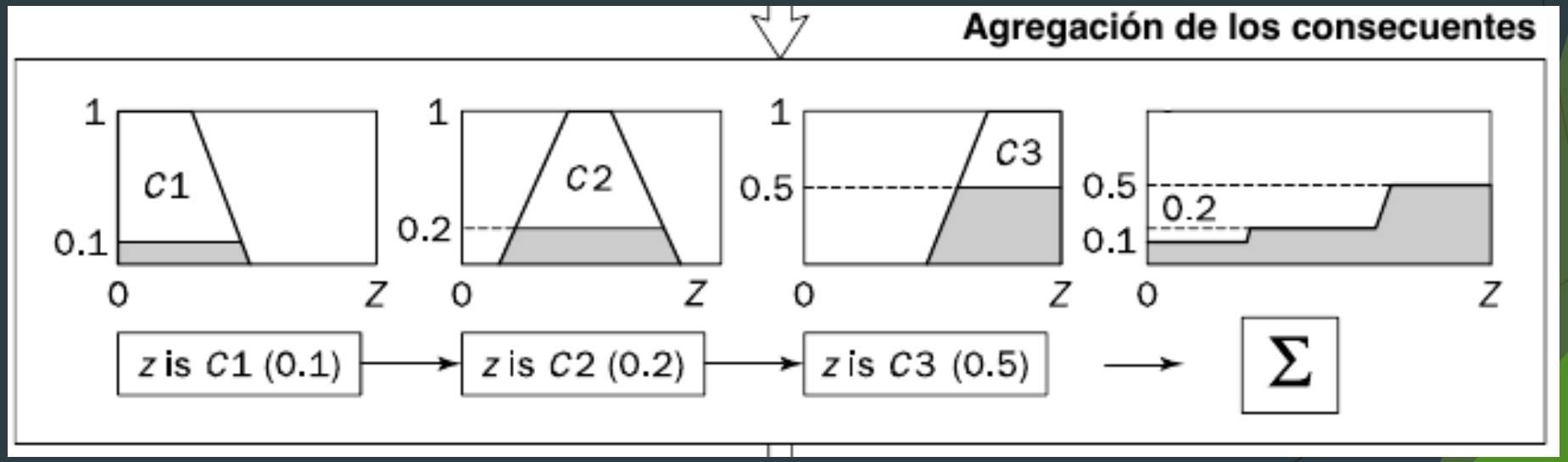
Sistemas expertos difusos

- Ejemplo de inferencia difusa.
  - **Etapa 2. Evaluación de las reglas**.



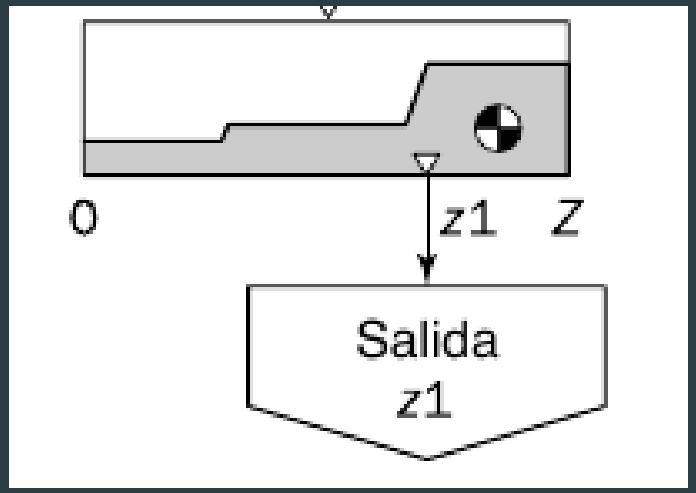
Sistemas expertos difusos

- Ejemplo de inferencia difusa.
  - **Etapa 3. Agregación de las salidas**.
    - Unificar las salidas de todas las reglas.
    - Combinar los consecuentes de todas las reglas una vez recortados o escalados.



Sistemas expertos difusos

- Ejemplo de inferencia difusa.
  - **Etapa 4. Defuzificación**.
    - El resultado requiere un valor no difuso.
    - Habitualmente, el resultado se calcula como el centroide del conjunto de salida agregado en el paso 3.



Razonamiento monotónico

- La mayoría de sistemas lógicos tienen una relación de consecuencia monótona.
  - Al agregar una fórmula a una teoría nunca se produce una reducción de su conjunto de consecuencias.
  - La monoticidad indica que al agregar nuevos conocimientos no se reduce el conjunto de cosas conocidas (lo que es verdad no deja de serlo).
  - El conjunto de cosas sabidas es siempre creciente.
- El razonamiento monótono es parte de la lógica clásica, abarcando las siguientes lógicas:
  - Lógica proposicional. Trata la verdad o falsedad de las proposiciones y cómo se transmite la verdad de las premisas a la conclusión.
  - Lógica de primer orden o lógica de predicados. Trata sobre los objetos y las relaciones entre ellos.
- El razonamiento deductivo es el tipo de razonamiento monótono por excelencia.

Razonamiento monotónico

- Una lógica monótona o monotónica no puede manejar ciertos tipos de razonamiento.
  - Razonamiento por defecto. Los hechos pueden ser conocidos únicamente por la carencia de evidencia de lo contrario.
  - Razonamiento abductivo. Los hechos sólo se deducen en calidad de explicaciones probables.
  - Razonamiento acerca del conocimiento. La ignorancia de un hecho debe ser retractada cuando el hecho sea conocido.
  - Revisión de creencias donde el nuevo conocimiento puede contradecir creencias anteriores, obligando a revisarlas.
- Las limitaciones del razonamiento monótono son un inconveniente en gran cantidad de problemas que se presentan en IA, que tienen carácter no monótono.
  - A medida que avanza el proceso de inferencias, nuevas evidencias o acciones del sistema anulan premisas o conclusiones anteriores.
  - Se requiere una lógica no monótona para formalizar esta anulación de premisas.
  - Un proceso frecuente en IA es el razonamiento por defecto: suponer que algo es verdadero o falso mientras no haya evidencia de lo contrario.

Razonamiento no monotónico

- Habitualmente, el conocimiento disponible es incompleto.
- Para modelar razonamiento de sentido común, es necesario obtener conclusiones plausibles a partir del conocimiento dado.
  - Para obtener conclusiones plausibles es necesario realizar asunciones.
  - La elección de estas asunciones no es aleatoria: la mayoría del conocimiento del mundo puede ser expresado a partir de reglas generales que especifican propiedades típicas de los objetos.
    - Ejemplo: “Los pájaros vuelan”. Los pájaros típicamente vuelan, aunque puede haber excepciones como los pingüinos o las avestruces.
- El razonamiento no monotónico trata el problema de derivar conclusiones plausibles, pero no infalibles, a partir de una base de conocimiento incompleto.
- Como las conclusiones no son certezas, se pueden retractar si nueva información indica que son incorrectas.

Razonamiento no monotónico

- Ejemplo:
  - La base de conocimiento contiene:
    - Típicamente los pájaros vuelan.
    - Los pingüinos no vuelan.
    - Piolín es un pájaro.
  - Es plausible concluir que Piolín vuela.
  - Sin embargo, añadimos la siguiente información a la base de conocimiento:
    - Piolín es un pingüino.
  - La conclusión previa debe ser retractada y, en su lugar, debe sustituirse por una nueva conclusión:
    - Piolín no vuela.

Razonamiento no monotónico

- La declaración “típicamente A” se puede interpretar como “en la ausencia de información de lo contrario, se asume A”.
- El problema es definir de manera precisa el significado de “en ausencia de información de lo contrario”.
- El significado puede ser: “no hay nada en la base de conocimiento que es inconsistente con la asunción A”.
- Existen otras interpretaciones, que pueden derivar en diferentes lógicas no monotónicas.

Razonamiento no monotónico

- La lógica clásica es inadecuada ya que es monotónica.
  - Si la fórmula B es derivable de un conjunto de fórmulas S, entonces B es también derivable de cualquier superconjunto de S.
    - $S \vdash B \rightarrow S \cup \{A\} \vdash B$, para cualquier fórmula A.
  - No se puede representar una regla como “típicamente los pájaros vuelan” como:
    - $\forall x(pajaro(x) \wedge \neg excepcion(x) \rightarrow volar(x))$
  - Y después añadir las excepciones:
    - $\forall x(excepcion(x) \leftrightarrow pingüino(x) \vee avestruz(x) \vee \ldots)$
  - **No conocemos todas las excepciones por anticipado**.
  - Para concluir que Piolín vuela tenemos que probar que “Piolín no es una excepción”:
    - $\neg pingüino(piolin), \neg avestruz(piolin), \ldots$
  - Sin embargo, lo que realmente queremos es probar que Piolín vuela porque no podemos concluir que es una excepción, no porque podemos probar que no es una excepción.

Asunción de Mundo Cerrado

- **Asunción de Mundo Abierto (OWA)**: asumimos que el conocimiento del mundo es incompleto y toda la información no explícitamente declarada se considera desconocida.
- Observación de Reiter: hay muchos más hechos negativos que positivos.
- **Asunción de Mundo Cerrado (CWA)**: los únicos hechos verdaderos son los que se pueden probar a partir de la base de conocimiento. El resto de sentencias atómicas son tomadas como falsas por defecto.
  - Si $BD \nvdash A$ entonces $BD \vdash_{CWA} \neg A$
  - Esta inferencia no es válida en lógica clásica.
- Una comprensión básica de la lógica de base de datos es que solamente la información positiva se representa explícitamente. La información negativa no se representa explícitamente.
- Si un hecho positivo no está presente en la base de datos (BD), se asume su negación.

Asunción de Mundo Cerrado

- Ejemplo: suponer una BD que contiene hechos de la forma “practica(persona, deporte)”:
  - practica(ana, tenis)
  - practica(juan, tenis)
  - practica(ana, esquí)
- Entonces, se puede inferir:
  - BD $\vdash_{CWA}$ ¬practicar(juan, esquí)
- Trivialmente, la Asunción de Mundo Cerrado es no monotónica, ya que añadir un hecho puede conducir a descartar la conclusión negativa:
  - BD $\cup$ {practicar(juan, esquí)} $\nvdash_{CWA}$ ¬practicar(juan, esquí)

Negación como principio de fallo

- La programación lógica es el uso de la lógica matemática para la programación de ordenador.
- Prolog es uno de los ejemplos más conocidos, basado en una secuencia de cláusulas definidas de Horn.
  - Una cláusula de Horn es una cláusula (disyunción de literales) con al menos un literal positivo.
    - Una claúsula de Horn con exactamente un literal positivo es una cláusula definida o cláusula de Horn estricta.
    - Una claúsula definida sin literales negativos es una cláusula unitaria.
    - Una cláusula unitaria sin variables es un hecho.
    - Una cláusula de Horn sin literales positivos es una cláusula objetivo.
    - La cláusula vacía, consistente en ningún literal (equivalente a falso) es una cláusula objetivo.
    - Si queremos probar φ, asumimos ¬φ (el objetivo) y comprobamos si esa asunción lleva a alguna contradicción. Si se da ese caso, entonces φ se cumple.

Negación como principio de fallo

Tipo de claúsula de Horn	Forma disyuntiva	Forma de implicación	Lectura intuitiva
Claúsula definida	$\neg p \vee \neg q \vee ... \vee \neg t \vee u$	$u \leftarrow p \wedge q \wedge ... \wedge t$	Asumimos que, si p y q y ... y t todos son verdad, entonces u se cumple
Hecho	u	$u \leftarrow$ verdad	Asumimos que u se cumple
Cláusula objetivo	$\neg p \vee \neg q \vee ... \vee \neg t$	falso $\leftarrow p \wedge q \wedge ... \wedge t$	Se demuestra que p y q y ... y t todos se cumplen

Negación como principio de fallo

- Prolog permite el uso de expresiones negativas mediante el operador especial “not”.
- Prolog no distingue entre no ser capaz de encontrar una derivación y afirmar que una consulta es falsa, no distingue entre “falso” y “desconocido”.
- El operador not es un predicado predefinido que se basa en el principio de **negación como principio de fallo (NAF)**.
- NAF es una regla no monotónica de inferencia usada para derivar ¬p a partir del fallo para derivar p.
  - Dado un literal positivo p, not(p) se evalúa a verdad si no se puede encontrar una prueba finita para p a partir de la información contenida en el sistema.
  - En otro caso, se toma su valor como falso.
- NAF y CWA tienen una idea común: si un literal positivo no puede ser “probado”, entonces su forma negativa puede ser interpretada como verdad.

Negación como principio de fallo

- Ejemplo:

  volar(x) :- pajaro(x), not(anomalo(x)).

  anomalo(x) :- avestruz(x).

  pajaro(piolin).

  - ?- anomalo(piolin).

    ---> falso
  - ?- volar(piolin).

    ---> verdad

Negación como principio de fallo

- Ejemplo (añadiendo información se puede modificar la salida):

  volar(x) :- pajaro(x), not(anomalo(x)).

  anomalo(x) :- avestruz(x).

  pajaro(piolin).

  avestruz(piolin).

  - ?- anomalo(piolin).

    ---> verdad
  - ?- volar(piolin).

    ---> falso

Problema del Marco

- El Problema del Marco, o Frame Problem, trata el problema de representar un mundo dinámico.
- ¿Cómo representar que los objetos no han sido afectados por un cambio de estado?
  - Ejemplo: Mover un objeto no cambia su color.
- En una representación basada en lógica clásica, se debe indicar explícitamente la persistencia de las propiedades de los objetos.
  - Se necesita una gran cantidad de axiomas de frame.
    - $\forall x \forall c \forall s \forall l(color(x, c, s) \rightarrow color(x, c, resultado(mover, x, l, s)))$
    - $\forall x \forall c \forall s(color(x, c, s) \rightarrow color(x, c, resultado(t\_luz\_encendida, s)))$
    - $\forall x \forall c \forall s(color(x, c, s) \rightarrow color(x, c, resultado(abrir\_puerta, s)))$
    - ...

Problema del Marco

- Necesitaríamos un meta-axioma general de la forma:
  - $\forall p \forall a \forall s(\text{mantener}(p, s) \wedge \neg \text{excepcion}(p, a, s) \rightarrow \text{mantener}(p, \text{resultado}(a, s)))$
- Sin embargo, necesitamos ser capaces de concluir que una acción no es una excepción a la preservación de una propiedad determinada, a menos que podamos demostrar que realmente lo es.
- Necesitamos un mecanismo de razonamiento no monotónico.
- La **circunscripción** es una lógica no monotónica que formaliza la asunción de sentido común de que las cosas son como se espera a menos que se especifique otra cosa.
  - Se trata de una formalización lógica del principio físico de inercia.

Lógica por defecto

- La lógica por defecto extiende la lógica clásica utilizando reglas de inferencia no estándar.
  - Ejemplo: $\frac{\text{pajaro}(x) : \text{volar}(x)}{\text{volar}(x)}$
    - Se puede interpretar como “si x es un pájaro y podemos asumir consistentemente que x vuela, entonces podemos inferir que x vuela”.
- De forma más general, tenemos reglas de la siguiente forma:

$$\frac{\alpha(x) : \beta(x)}{\gamma(x)}$$

  - Se puede interpretar como “si α(x) se cumple y β(x) se puede asumir consistentemente, entonces podemos concluir γ(x)”.
  - Terminología:
    - α(x): prerrequisito.
    - β(x): justificación.
    - γ(x): consecuente.

Teoría por defecto

- Una teoría por defecto es un par <D,W>, donde D es un conjunto de reglas por defecto y W es un conjunto de fórmulas de primer orden.
- Intuitivamente, en una teoría por defecto <D, W>:
  - *W* representa el conocimiento estable (pero incompleto) del mundo.
  - *D* representa las reglas para extender el conocimiento *W* por conclusiones plausibles (pero anulables).
  - Noción de **extensión de una teoría por defecto**: la teoría (conjunto deductivamente cerrado de fórmulas lógicas) obtenido mediante la extensión de *W* con las reglas de *D*.
- Ejemplo: *<D, W>*
  - D = { $\frac{\text{pajaro(x) : volar(x)}}{\text{volar(x)}}$ }
  - W = { pajaro(piolin), ∀x(pingüino(x) → pajaro(x)), ∀x(pingüino(x) → ¬volar(x)) }
    - Como pajaro(piolin) es verdadero, y es consistente asumir que volar(piolin), entonces volar(piolin) es verdad en la única extensión de <D,W>
    - Si consideramos la teoría <D, W'> donde W' = W ∪ {pingüino(piolin)}, entonces la asunción volar(piolin) ya no es consistente, y la aplicación de la regla por defecto se bloquea.

Teoría por defecto

- Ejemplo 2: <D, W>
  - D = {$d_1 = \frac{\text{mamifero}(x) : \neg\text{oviparo}(x)}{\neg\text{oviparo}(x)}$, $d_2 = \frac{\text{tener\_pico}(x) : \text{oviparo}(x)}{\text{oviparo}(x)}$ }
  - W = {mamifero(ornitorrinco), tener_pico(ornitorrinco)}
    - Para ambas reglas por defecto d_i, el prerrequisito es derivable a partir de W.
    - Si aplicamos d_1, podemos concluir ¬oviparo(ornitorrinco). Sin embargo, como oviparo(ornitorrinco) ya no se puede asumir consistentemente, entonces d_2 se bloquea.
    - Si aplicamos d_2, podemos concluir oviparo(ornitorrinco). Sin embargo, como ¬oviparo(ornitorrinco) no se puede asumir consistentemente, entonces d_1 se bloquea.
  - Hay dos extensiones, una conteniendo ¬oviparo(ornitorrinco) y otra con oviparo(ornitorrinco).
  - Una extensión representa el conjunto de conclusiones plausibles.
  - Una teoría por defecto puede tener 0, 1 o más extensiones.

Extensiones

- Dada una teoría por defecto Δ=<D,W>, un conjunto de fórmulas E es una extensión de Δ si:
  - E es deductivamente cerrado: E = Th(E). Todas las fórmulas de E se pueden deducir de forma lógica a partir de ellas. $Th(S) = \{C \in L \mid S \vdash C\}$
  - Todas las reglas por defecto con respecto a E han sido aplicadas.
    - Para todo $\frac{\alpha : \beta}{\gamma} \in D$, si $\alpha \in E$ y $\neg\beta \notin E$ entonces $\gamma \in E$
  - Definición semi-inductiva:
    - $S_0 = W$
    - $S_{i+1} = Th(S_i) \cup \{\gamma \mid \frac{\alpha : \beta}{\gamma} \in D, \alpha \in Si, \neg\beta \notin E\}$
    - $E = \cup_i S_i$
  - El orden en el que las reglas por defecto en cada etapa S_{i+1} es significativo: diferentes ordenaciones pueden llevar a diferentes extensiones y, por tanto, diferentes conclusiones.

Extensiones

- Ejemplo 1: $\Delta = \{b, p \rightarrow \neg f\}, \frac{\{b : f\}}{f}$
  - Única extensión: $E = Th(\{b, p \rightarrow \neg f, f\})$
    - $S_0 = \{b, p \rightarrow \neg f\}$
    - $S_1 = S_0 \cup \{f\}$, ya que $S_0 \vdash b$ y $\neg f \notin E$
- Ejemplo 1': $\Delta = \{b, p \rightarrow \neg f, p\}, \frac{\{b : f\}}{f}$
  - Única extensión: $E = Th(\{b, p \rightarrow \neg f, p\})$
    - $S_0 = \{b, p \rightarrow \neg f, p\}$
    - $S_1 = S_0$, ya que $S_0 \vdash b$ pero $\neg f \in E$

Extensiones

- Ejemplo 2: $\Delta = \{r, q\}, \{d1 = \frac{r : \neg p}{\neg p}, d2 = \frac{q : p}{p}\}$
  - Extensión E_1: $E_1 = Th(\{r, q, \neg p\})$
    - $S_0 = \{r, q\}$
    - $S_1 = S_0 \cup \{\neg p\}$, mediante la aplicación de d_1 ya que $S_0 \vdash r$ y $\neg\neg p \notin E$
    - $S_2 = S_1$ ya que d_2 no se puede aplicar porque $\neg p \in E$
  - Extensión E_2: $E_2 = Th(\{r, q, p\})$
    - $S_0 = \{r, q\}$
    - $S_1 = S_0 \cup \{p\}$, mediante la aplicación de d_2 ya que $S_0 \vdash q$ y $\neg p \notin E$
    - $S_2 = S_1$ ya que d_1 no se puede aplicar porque $\neg\neg p \in E$

Extensiones

- Ejemplo 3: $\Delta = \{\emptyset\}, \{d1 = \frac{: a}{\neg a}\}$
  - Si $\neg a \notin E$, entonces deberíamos aplicar la regla d1 y, entonces, $\neg a \in E$.
  - Sin embargo, si $\neg a \in E$, entonces la regla por defecto es inaplicable, por lo que deberíamos tener $\neg a \notin E$.
  - Por tanto, Δ no tiene extensiones válidas.

Extensiones

- Ejemplo 4: $\Delta = \{\emptyset\}, \{d1 = \frac{: \neg p}{q}, d2 = \frac{: \neg q}{p}\}$
  - Extensión E_1: $E_1 = Th(\{q\})$
    - $S_0 = \emptyset$
    - $S_1 = S_0 \cup \{q\}$, mediante la aplicación de d_1 ya que $\neg\neg p \notin E$
    - $S_2 = S_1$ ya que d_2 no se puede aplicar porque $\neg\neg q \in E$
  - Extensión E_2: $E_2 = Th(\{p\})$
    - $S_0 = \emptyset$
    - $S_1 = S_0 \cup \{p\}$, mediante la aplicación de d_2 ya que $\neg\neg q \notin E$
    - $S_2 = S_1$ ya que d_1 no se puede aplicar porque $\neg\neg p \in E$

Extensiones

- Ejemplo 5: $\Delta = \{\emptyset\}, \{d1 = \frac{a : b}{b}, d2 = \frac{b : a}{a}\}$
  - Extensión E = Th($\emptyset$)
    - $S_0 = \emptyset$
    - $S_1 = S_0$ ya que $S_0 \nvdash a$, y $S_0 \nvdash b$

Teoría por defecto normal

- Una regla por defecto d es normal si tiene la forma $\frac{\alpha : \beta}{\beta}$
- Una teoría por defecto normal $\Delta =< W, D >$ es una teoría por defecto donde todas las reglas por defecto en D son normales.
- Una teoría por defecto normal siempre tiene una extensión.

Relación de inferencia

- Como a partir de una teoría por defecto $\Delta =< W, D >$ se pueden tener múltiples extensiones (incluso ninguna), es necesario definir la noción de inferencia.
  - **Inferencia crédula**. $\Delta \vdash_c A$ si existe una extensión E de Δ de forma que $A \in E$.
    - En esta modalidad de inferencia, basta con que exista una extensión que contenga el hecho A para que sea probada.
  - **Inferencia escéptica**. $\Delta \vdash_s A$ si para todas las extensiones E de Δ, tenemos que $A \in E$.
    - En esta modalidad de inferencia, A debe aparecer en todas las extensiones para que sea probada.
  - Como una teoría por defecto puede no tener extensiones, $\Delta \vdash_s A$ no implica $\Delta \vdash_c A$.

Problemas con la lógica por defecto

- Se puede producir una **transitividad no deseada**.
  - Δ =< W, D >, W = {estudiante}, D = {d1 = $\frac{\text{estudiante : adulto}}{\text{adulto}}$, d2 = $\frac{\text{adulto : trabajar}}{\text{trabajar}}$}
    - Δ tiene una extensión única E = {estudiante, trabajar, adulto}
    - Se trata de una extensión contraintuitiva (los estudiantes no suelen trabajar)
    - Si añadimos la regla por defecto $\frac{\text{estudiante : ¬trabajar}}{\text{¬trabajar}}$, la teoría pasa a tener dos extensiones:
      - E1 = {estudiante, adulto, trabajar}
      - E2 = {estudiante, adulto, ¬trabajar}. Esta teoría sería más plausible.
    - Solución: reemplazar d2 por d2 = $\frac{\text{adulto : trabajar ^ ¬estudiante}}{\text{trabajar}}$
      - La única extensión es E2 = {estudiante, adulto, ¬trabajar}
      - La regla por defecto no es normal sino seminormal, la justificación implica el consecuente.
      - Una teoría por defecto seminormal (donde todas las reglas por defecto son seminormales) puede no tener extensiones.

Problemas con la lógica por defecto

- Existen problemas manejando la especificidad.
  - $\Delta =< W, D >$, W = {usuario, baneado}, D = {d1 = $\frac{\text{usuario} : \text{login}}{\text{login}}$, d2 = $\frac{\text{usuario} \wedge \text{baneado} : \neg\text{login}}{\neg\text{login}}$}
    - La teoría tiene dos extensiones:
      - E1 = {usuario, baneado, login}
      - E2 = {usuario, baneado, ¬login}. Esta teoría es la deseada.
    - Solución: asignar prioridad a las reglas por defecto en base a su especificidad. Este orden de prioridad es tenido en cuenta para calcular las extensiones.