Inteligencia Artificial II

Trabajo Práctico N°2: Razonamiento

Francisco Castel

Melanie Martinez

Brian Nuñez

Juan Ignacio Quiroga

Nicolas Sorrentino

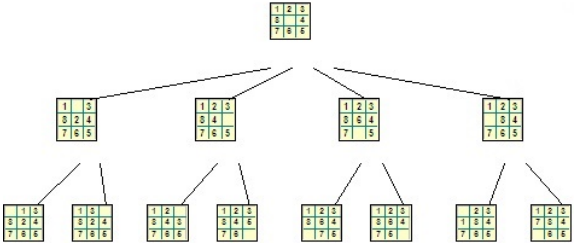
Universidad Nacional de Cuyo

Contenido

1. Algoritmos Basados en Inferencia Lógica
2. Lógica Proposicional
3. Lógica de Primer Orden
4. Planteamiento del Problema
5. Diseño del Sistema
6. Conjuntos Difusos
7. Resultados de la Simulación

¿Por qué usar Alg. Inferencia Lógica en lugar de Alg. de Búsqueda?

- Los algoritmos de búsqueda son útiles cuando el espacio de estados está bien definido y es finito.



¿Por qué usar Alg. Inferencia Lógica en lugar de Alg. de Búsqueda?

- Los algoritmos de inferencia lógica permiten razonamientos explícitos y adaptativos a partir del conocimiento disponible.
- **Ventajas de la inferencia lógica:**

Expresa declarativamente el conocimiento: lo cual lo hace más escalable y flexible.
Inferencia a partir de información incompleta: Puede generar conclusiones con información parcial.
Reutilización del conocimiento: La misma base de conocimientos puede ser utilizada para diferentes tareas.

¿Por qué usar Alg. Inferencia Lógica en lugar de Alg. de Búsqueda?

- **Desventajas de la inferencia lógica:**

Aun que la expresividad sea más flexible. Este sigue siendo limitada para representar el conomiento general.

Imposibilidad de razonar de formas no previstas: Solo opera dentro de lo que está explicitamente definido.

Dificultad de alterar comportamientos dinamicamente: No se puede incorporar nueva informacion inesperada sin reconfiguración.

¿Por qué usar Alg. Inferencia Lógica en lugar de Alg. de Búsqueda?

Ejemplo práctico: Un sistema experto médico, en el cual se pueden representar sintomas y enfermedades mediantes reglas como:

- **Si el paciente tiene fibre y tos, entonces podrias tener gripe.**
- **si el paciente tiene erupciones cutáneas y fiebre, entonces podría tener sarampión.**

A partir de los sintomas observados, el sistema puede inferir posibles diagnosticos, incluso si no se presentan todos los sitomas tipicos de una enfemedad.

Base de Conocimiento

Existen dos grandes tipos de conocimientos, que nosotros vamos a incorporar a una base de conocimientos;

Background knowledge (KB)	**Ground Truth**
Contiene hechos, reglas y relaciones del dominio que es utilizada para razonar y deducir nueva información. Esto se puede incorporar mediante; • **Conocimiento incorporado:** Es lo que modela un ingeniero en conocimiento. • **Aprendizaje Automatico:** Una vez que empieza a operar uno puede incorporar aprendizaje adicional.	Son hechos que son sensados de forma automatica por el agente o incorporardos de forma explicita a traves de un medio de comunicación con el agente.

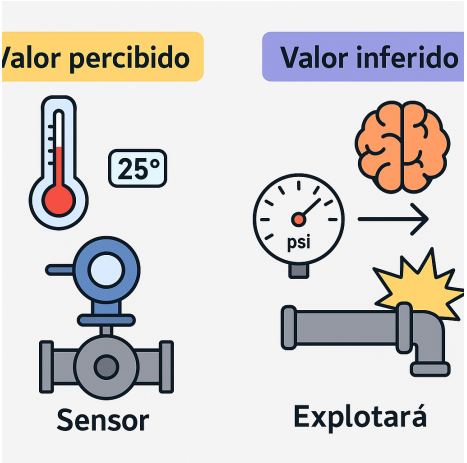
¿Qué es un valor inferido y un valor percibido?

Valor Percibido

Es un dato obtenido directamente de un sensor (por ejemplo, la temperatura medida por un termómetro).

Valor Inferido

Es un dato calculado a partir de hechos y reglas usando inferencia lógica.



¿Qué valores puede tomar una oración atómica de acuerdo a la lógica proposicional?

- Una oración atómica puede tomar solo dos valores: **Verdadero** o **Falso**.
- En lógica proposicional, las oraciones son estrictamente dicotómicas.
- Se representan mediante símbolos como P, Q, R, que se pueden combinar en sentencias complejas.

$$
\begin{array}{rcl}
\textit{Sentencia} & \rightarrow & \textit{Sentencia Atómica} \mid \textit{Sentencia Compleja} \\
\textit{Sentencia Atómica} & \rightarrow & \textbf{Verdadero} \mid \textbf{Falso} \mid \textit{Símbolo Proposicional} \\
\textit{Símbolo Proposicional} & \rightarrow & \textbf{P} \mid \textbf{Q} \mid \textbf{R} \mid \ldots \\
\textit{Sentencia Compleja} & \rightarrow & \neg\, \textit{Sentencia} \\
 & \mid & (\textit{Sentencia} \wedge \textit{Sentencia}) \\
 & \mid & (\textit{Sentencia} \vee \textit{Sentencia}) \\
 & \mid & (\textit{Sentencia} \Rightarrow \textit{Sentencia}) \\
 & \mid & (\textit{Sentencia} \Leftrightarrow \textit{Sentencia})
\end{array}
$$

Modelado en Lógica Proposicional

- Para modelar la frase *"Los valores de A comprendidos entre 0 y 9"*, debemos:
  - Definir el dominio: $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$
  - Asignar oraciones atómicas:
    - A_0: "A es 0"
    - A_1: "A es 1"
    - $\vdots$
    - A_9: "A es 9"
  - Unir las oraciones atómicas con disyunciones lógicas:

$$A_0 \vee A_1 \vee A_2 \vee A_3 \vee A_4 \vee A_5 \vee A_6 \vee A_7 \vee A_8 \vee A_9$$

- Esto indica que al menos uno de los valores de A en el rango de 0 a 9 debe ser verdadero.

¿Qué algoritmos se utilizan para resolver problemas modelados con lógica proposicional?

Algoritmos de Inferencia

Permite obtener conclusiones lógicas a partir de las proposiciones iniciales que tenemos.

Algoritmos de Satisfacibilidad

Los algoritmos de satisfacibilidad verifican si una proposición lógica es satisfacible, es decir, si existe una asignación de valores de verdad que haga que la proposición completa sea verdadera.

Algoritmos de Inferencia

- **Encadenamiento (Forward Chaining y Backward Chaining):**En el encadenamiento hacia adelante, se parte de los hechos conocidos para inferir nuevas conclusiones, mientras que en el encadenamiento hacia atrás se empieza por la meta o conclusión deseada y se buscan las premisas que la soporten.
- **Método de Resolución:** Al aplicar este método, primero transformamos la fórmula a forma normal conjuntiva (FNC), luego resolvemos las cláusulas con literales opuestos, y finalmente, si obtenemos una contradicción (la cláusula vacía), concluimos que la fórmula es insatisfacible.

Algoritmos de Satisfacibilidad

- **DPLL (Davis-Putnam-Logemann-Loveland):** Sirve para comprobar la satisfacibilidad de una fórmula booleana. Utiliza backtracking (retroceso) y técnicas como la propagación unitaria para reducir el espacio de búsqueda.
- **WalkSAT** Es un algoritmo de búsqueda local para encontrar una asignación satisfactoria de valores de verdad en un conjunto de proposiciones booleanas.

¿Qué valores puede tomar una oración atómica de acuerdo a la lógica de primer orden?

$$
\begin{array}{rcl}
\mathit{Sentencia} & \rightarrow & \mathit{SentenciaAtómica} \\
 & | & (\mathit{Sentencia\ Conectiva\ Sentencia}) \\
 & | & \mathit{Cuantificador\ Variable} \ldots \mathit{Sentencia} \\
 & | & \neg \mathit{Sentencia} \\
 & & \\
\mathit{SentenciaAtómica} & \rightarrow & \mathit{Predicado}(\mathit{Término} \ldots) \mid \mathit{Término} = \mathit{Término} \\
 & & \\
\mathit{Término} & \rightarrow & \mathit{Función}(\mathit{Término} \ldots) \\
 & | & \mathit{Constante} \\
 & | & \mathit{Variable} \\
 & & \\
\mathit{Conectiva} & \rightarrow & \Rightarrow \mid \wedge \mid \vee \mid \Leftrightarrow \\
\mathit{Cuantificador} & \rightarrow & \forall \mid \exists \\
\mathit{Constante} & \rightarrow & A \mid X_1 \mid \mathit{Juan} \mid \cdots \\
\mathit{Variable} & \rightarrow & a \mid x \mid s \mid \cdots \\
\mathit{Predicado} & \rightarrow & \mathit{AntesDe} \mid \mathit{TieneColor} \mid \mathit{EstáLLoviendo} \mid \cdots \\
\mathit{Función} & \rightarrow & \mathit{Madre} \mid \mathit{PiernaIzquierda} \mid \cdots
\end{array}
$$

Lógica de primer orden

Es una extensión de la lógica proposicional que permite expresar relaciones complejas entre objetos y propiedades dentro de un dominio. A diferencia de la lógica proposicional, que solo maneja proposiciones completas, la lógica de primer orden introduce predicados, funciones y cuantificadores, lo que le otorga mayor expresividad.

Modelado de A comprendido entre 0 y 9 en LPO

- La frase *"Los valores de A comprendidos entre 0 y 9"* se puede modelar en LPO como:

$$\exists x(0 \leq x \leq 9)$$

Algoritmos en Lógica de Primer Orden

- **Encadenamiento hacia delante:** Data-driven, inferencias impulsadas por hechos.
- **Encadenamiento hacia atrás:** Goal-driven, inferencias guiadas por metas.
- **Resolución:** Prueba por contradicción sobre cláusulas en CNF.

LPO vs LP

- **Variables:** símbolos como `X`, `S` que pueden tomar distintos valores (p.ej. nombres de sensores, usuarios, objetos).
- **Predicados:** relaciones genéricas o propiedades que usan variables, p.ej. `sensorActivo(S)` o `temperatura(S,T)` en lugar de hechos atómicos fijos.

Unificación:

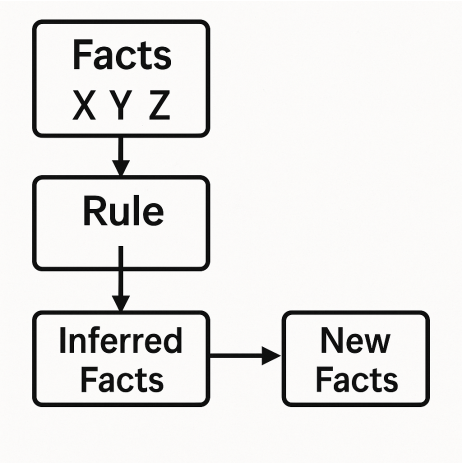
- Opera sustituyendo variables para que dos términos coincidan.
- Ejemplo: para $f(X, a)$ y $f(b, Y)$ la unificación es

$$X = b, \quad Y = a.$$

- Se emplea en forward chaining, backward chaining y resolución, permitiendo instanciar reglas y metas de forma genérica.

Forward Chaining en LPO

- Proceso impulsado por los **datos**: parte de los hechos conocidos
- **Correcto**: nunca infiere algo que no esté lógicamente justificado.
- **Completo para cláusulas de Horn**: encuentra *todas* las consecuencias lógicas posibles; ninguna conclusión verdadera queda fuera.
- **Data–driven**: el motor de reglas solo se inicia cuando entra o cambia un hecho.
- **Uso típico**: sistemas reactivos e IoT (OPS5, CLIPS, Drools).



Mecánica general

- **Unificación:** empareja variables con términos para que una regla "encaje" con un hecho.
- **Instanciación:** tras unificar, generamos un nuevo hecho concreto (*ground fact*).
- **Ejemplo (ingeniería):**

$$\text{Regla: } alarma(S) \ :- \ sensorActivo(S),\ temperatura(S,T),\ T > 50.$$
$$\text{Hechos: } sensorActivo(term1),\ temperatura(term1, 55).$$
$$\text{Unificación: } S = term1,\ T = 55 \quad \Longrightarrow \quad alarma(term1).$$

- Repetir hasta que no queden reglas aplicables o se alcance el objetivo.

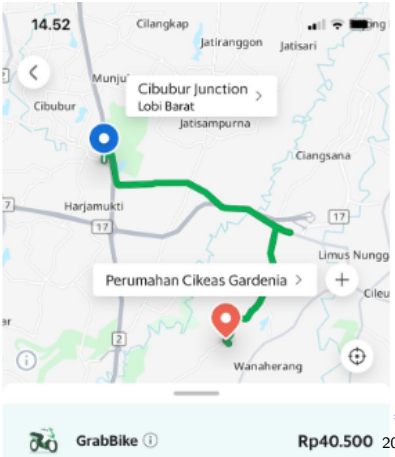
Ejemplo de aplicación

- Eventos en tiempo real: solicitudes, ubicación, tráfico.
- Cada hecho dispara el forward chaining.
- Se generan nuevos hechos automáticamente.

Cada hecho nuevo (p. ej. "lluvia intensa", "demanda_alta(zona A)") activa reglas como

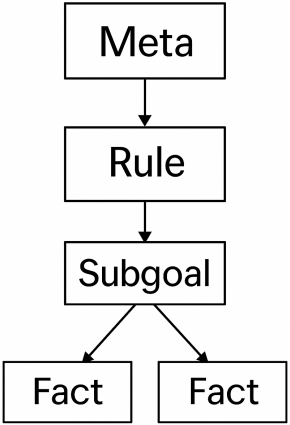
$$\text{lluvia intensa} \wedge \text{hora pico} \longrightarrow \text{activar tarifa dinámica.}$$

El consecuente (activar tarifa dinámica) se añade como hecho y puede disparar reglas posteriores: reasignación de flota, recomendación de GrabBike, etc.



Backward Chaining en LPO

- Proceso impulsado por los **objetivos**.
- **Goal–driven**: el razonamiento se activa por la consulta, no por datos.
- **Horn–driven**: unifica la meta con la cabeza de cláusulas de Horn para generar sub-metas.
- Prueba recursivamente cada sub-meta contra hechos o reglas.
- Uso típico en sistemas expertos y lenguajes lógicos (p. ej. Prolog).



Mecánica de Backward Chaining (Prolog)

- **Consulta inicial:**

```
?- parar_ventilador.
```

- **Selecciona regla:**

```
parar_ventilador :- sobrecalentado.
```

- **Genera sub-meta:**

```
?- sobrecalentado.
```

- **Descompone regla:**

```
sobrecalentado :- temp_motor(T), T > 90.
```

- **Evalúa sub-metas:**

```
?- temp_motor(75).     % hecho -> true
?- 75 > 90.            % aritmética -> false
```

- **Backtracking:** Probar otras reglas o hechos; si no hay más, la consulta falla.

Ejemplo de aplicación

Backward Chaining en Prolog

- Goal–driven: inicia con una consulta.
- Unifica meta con reglas Horn.
- Backtracking hasta éxito/fallo.

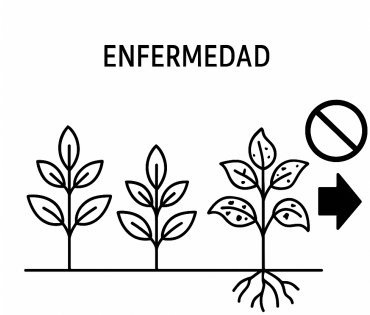
```
%hallar el maximo de una lista
%maxlista([7,3,-1,-100,500,3,5,40],S).
maxlista([H],H).
maxlista([H|T],N):- maxlista(T,NT), N  is max(NT,H).

%hallar el minimo de una lista
%minlista([7,3,-1,-100,500,3,5,40],S).
minlista([H],H).
minlista([H|T],N):- minlista(T,NT), N is min(NT,H).

%invertir una lista
%invlista( [8,9,7,2,1],S).
invlista([],[]).
invlista( [X|T],N):-invlista( [X|T],[],N).
invlista([],Zs,Zs).
invlista([X|Xs],Ys,Zs):- invlista(Xs,[X|Ys],Zs).
```

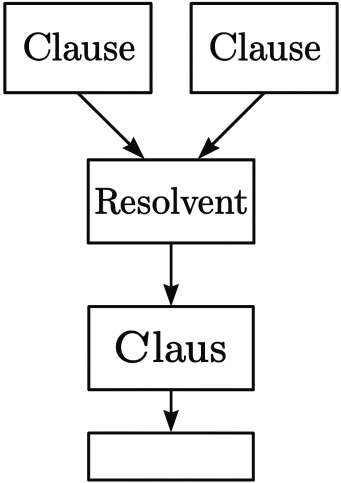
Sistema Vrikshayurveda

- Diagnóstico orientado a metas.
- Reglas tradicionales (Vrikshayurveda).
- Solo pregunta síntomas clave.



Algoritmo de Resolución en LPO

- Método de **prueba por contradicción**: añade la negación de la meta.
- Trabaja sobre cláusulas en **Forma Normal Conjuntiva** (CNF).
- Repite la **regla de resolución**: combina literales complementarios.
- Deriva la cláusula vacía ($\perp$) contradicción meta demostrada.



Ejemplo de Resolución: Robot y Objeto (1/2)

Escenario:

$$\forall r, o \ \big(DentroAlcance(r, o) \wedge PinzaAbierta(r) \ \rightarrow \ PuedeAgarrar(r, o)\big).$$

Hechos:

$$DentroAlcance(R1, O1), \quad PinzaAbierta(R1).$$

Meta:

$$PuedeAgarrar(R1, O1).$$

Cláusulas (CNF):

C1 $\neg DentroAlcance(r, o) \ \vee \ \neg PinzaAbierta(r) \ \vee$
$PuedeAgarrar(r, o)$

C2 $DentroAlcance(R1, O1)$

C3 $PinzaAbierta(R1)$

C4 $\neg PuedeAgarrar(R1, O1)$ (Negación de la meta)

Ejemplo de Resolución: Robot y Objeto (2/2)

Pasos de resolución:

1. Resolver C1 con C2 unificando $\{r = R1,\ o = O1\}$:

   $$(\neg DentroAlcance(r, o) \vee \neg PinzaAbierta(r) \vee PuedeAgarrar(r, o)),\ (DentroAlcance(R1, O1))$$

2. Resolver C5 con C3 sobre $PinzaAbierta(R1)$:

   $$(\neg PinzaAbierta(R1) \vee PuedeAgarrar(R1, O1)),\ (PinzaAbierta(R1)) \longrightarrow C6:\ PuedeAgarrar$$

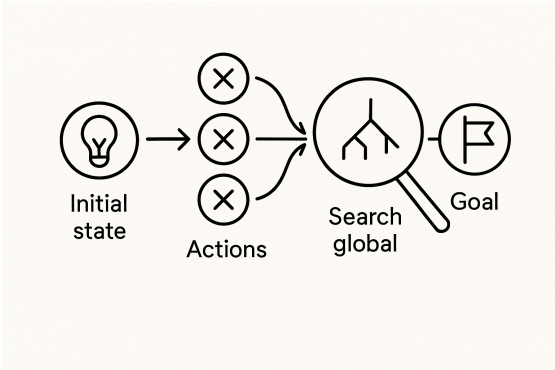
3. Resolver C6 con C4 sobre $PuedeAgarrar(R1, O1)$:

   $$(PuedeAgarrar(R1, O1)),\ (\neg PuedeAgarrar(R1, O1)) \longrightarrow \perp \quad \text{(cláusula vacía)}$$

Interpretación: La aparición de $\perp$ implica que la suposición $\neg PuedeAgarrar(R1, O1)$ contradice la KB. Por tanto, $PuedeAgarrar(R1, O1)$ queda demostrado.

Planificación: Definición y Ejemplo

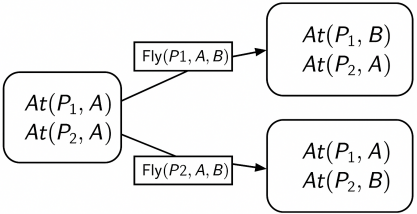
- Generar una secuencia de acciones que transforme un **estado inicial** en un **estado objetivo**.
- Basarse en el **conocimiento del dominio**: acciones con precondiciones y efectos.



- **Estado inicial:** $\{ Luz(Encendida) \}$
- **Acción:** `ApagarInterruptor()`
  - PRE: $Luz(Encendida)$
  - EFF: $Luz(Apagada)$
- **Objetivo:** $\{ Luz(Apagada) \}$

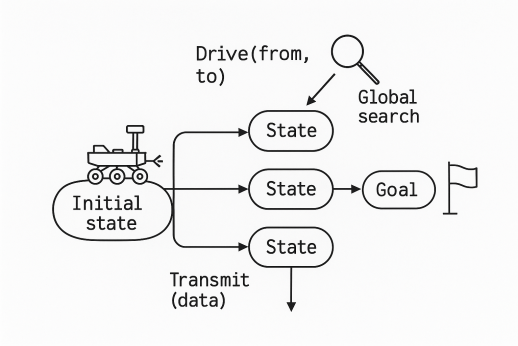
Planificación con Búsqueda Global

- Desde un **estado inicial**, el planner genera *sucesores* aplicando todas las acciones válidas.
- Aplica A* con $f(n) = g(n) + h(n)$ para expandir primero el estado de menor coste total y garantizar optimalidad.
- En cada nodo realiza el *goal test*: verifica si los literales objetivo están presentes.



Aplicación: Planning en Rovers de la NASA

- En misiones MER, cada sol se programan decenas de actividades científicas y de ingeniería.
- MAPGEN/EUROPA combina edición humana con búsqueda global basada en restricciones.
- Resuelve la coordinación de energía, memoria, temperatura y ventanas de comunicación.
- Garantiza planes factibles y óptimos en tiempo limitado.



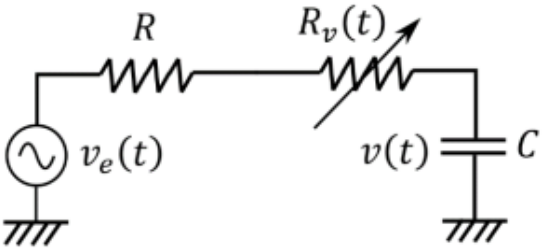
Planteamiento del Problema

Objetivo

Implementar un sistema de inferencia difusa para controlar la temperatura de una habitación.

- El sistema busca mantener la temperatura de confort (25°C) durante el día (8:00-20:00)
- La única variable controlable es la apertura de una ventana
- Se conocen las temperaturas interior y exterior, y se tiene pronóstico para 24h
- La ventana modifica la resistencia térmica entre interior y exterior

Modelo Térmico del Sistema



$$C\dot{v}(t) = \frac{v_e(t) - v(t)}{R + R_v(t)} \tag{1}$$

$$\dot{V}(t) = 2(v(t) - v_0)\dot{v}(t) \tag{2}$$

$$= 2\frac{(v(t) - v_0)(v_e(t) - v(t))}{C(R + R_v(t))} \tag{3}$$

Variable $z(t)$:

$$z(t) = (v(t) - v_0)(v_e(t) - v(t)) \tag{4}$$

- $R_v(t) \in [0, R_{vmax}]$, con $R_{vmax} = 0{,}1R$
- $v_0 = 25$ V (día)
- $v_{0_{CAL}} = 50$ V (noche, calentamiento)
- $v_{0_{ENF}} = 5$ V (noche, enfriamiento)

- Si $z > 0 \rightarrow$ Cerrar ventana
- Si $z \approx 0 \rightarrow$ Ventana en posición intermedia
- Si $z < 0 \rightarrow$ Abrir ventana

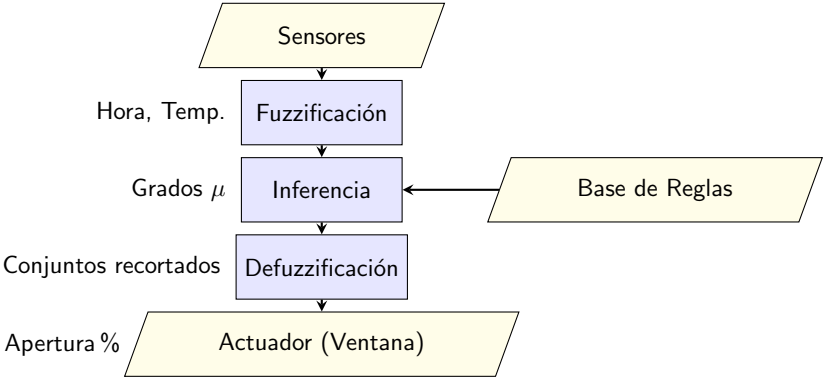
Base de Conocimientos

Reglas para el Control Difuso

Antecedente	**Consecuente**
HORA is DÍA $\wedge$ Z is POSITIVO	VENTANA is CERRAR
HORA is DÍA $\wedge$ Z is ZERO	VENTANA is CENTRO
HORA is DÍA $\wedge$ Z is NEGATIVO	VENTANA is ABRIR
HORA is NOCHE $\wedge$ $T_{PREDICHA}$ is ALTA $\wedge$ z_{ENF} is POSITIVO	VENTANA is CERRAR
HORA is NOCHE $\wedge$ $T_{PREDICHA}$ is ALTA $\wedge$ z_{ENF} is ZERO	VENTANA is CENTRO
HORA is NOCHE $\wedge$ $T_{PREDICHA}$ is ALTA $\wedge$ z_{ENF} is NEGATIVO	VENTANA is ABRIR
...	...

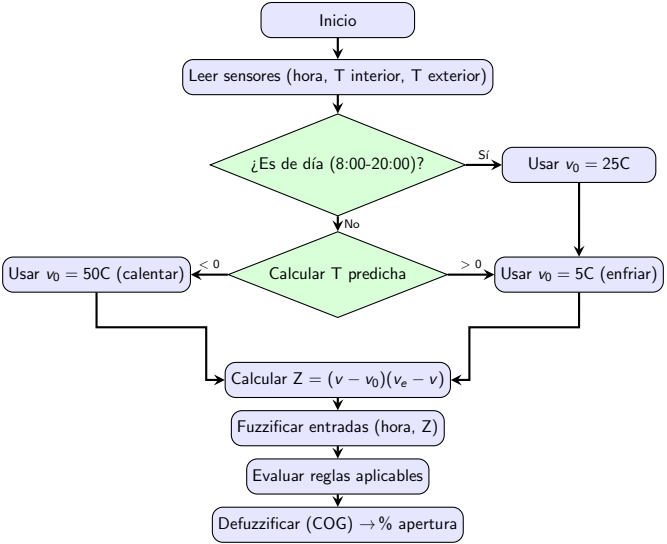
- Durante el día: Control basado en $z = (v(t) - v_0)(v_e(t) - v(t))$
- Durante la noche: Estrategia basada en predicción del día siguiente
  - Si se prevé calor: Enfriar ($v_0 = 5°C$)
  - Si se prevé frío: Calentar ($v_0 = 50°C$)

Arquitectura del Sistema



- El sistema sigue la arquitectura clásica de un controlador difuso:
  - Entrada $\rightarrow$ Fuzzificación $\rightarrow$ Inferencia $\rightarrow$ Defuzzificación $\rightarrow$ Salida
  - La base de reglas define el comportamiento del sistema

Diagrama de Flujo del Sistema



Estructura Orientada a Objetos del Sistema

Módulo	Clases principales	Responsabilidad
membership_functions	MembershipFunction, TriangularMF, TrapezoidalMF	Definir y evaluar funciones de pertenencia
fuzzy_set	FuzzySet	Agrupar una MF con nombre
linguistic_variable	LinguisticVariable	Colección de FuzzySets y universo de discurso
rule	FuzzyRule	Representar reglas "IF ... THEN ..."
rule_base	RuleBase	Conjunto de reglas; carga/validación
inference	FuzzyInferenceSystem	Fuzzificación, aplicación de reglas (min/max)
defuzzifier	Defuzzifier	Convertir salida borrosa a valor nítido
environment_controller	Environment, FuzzyController	Lectura de series, cálculo de z, orquestación del sistema

Estructura Orientada a Objetos del Sistema

MembershipFunction

- Clase abstracta base
- Implementaciones:
  - TriangularMF
  - TrapezoidalMF
  - SingletonMF
- Método (x) retorna grado de pertenencia

FuzzySet

- Representa conjunto borroso
- Asocia nombre con MF
- Ej: "POSITIVO", "NEGATIVO", "CENTRO"

LinguisticVariable

- Colección de conjuntos borrosos
- Define universo de discurso [min, max]
- Método fuzzify(x) calcula pertenencia a todos los conjuntos

FuzzyRule y RuleBase

- FuzzyRule: Antecedentes $\rightarrow$ Consecuente
- RuleBase: Colección de reglas
- Método evaluate() calcula fuerza de activación

Estructura Orientada a Objetos del Sistema

FuzzyInferenceSystem

- Implementa motor de inferencia Mamdani
- Fuzzifica entradas nítidas
- Evalúa reglas aplicables
- Calcula implicación (min)
- Agrega salidas (max)

Defuzzifier

- Métodos: COG, SOM, MOM, LOM, WAM
- Centro de Gravedad (COG)
- Centro de Máximos (MOM)
- Convierte conjunto borroso en valor nítido

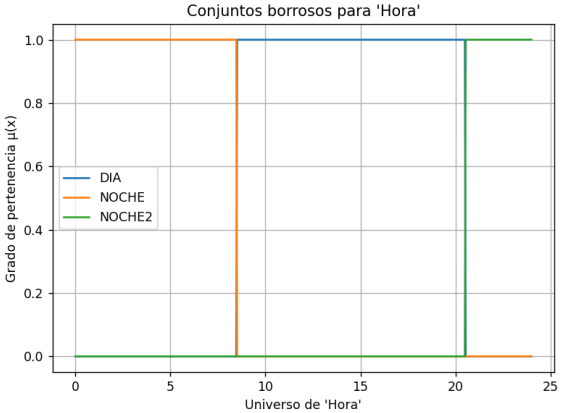
Environment

- Representa entorno físico
- Lee sensores (hora, temp. interior, exterior)
- Calcula variable Z: $(v - v_0)(v_e - v)$
- Calcula temperatura predicha (promedio)
- Adapta v_0 según condiciones

FuzzyController

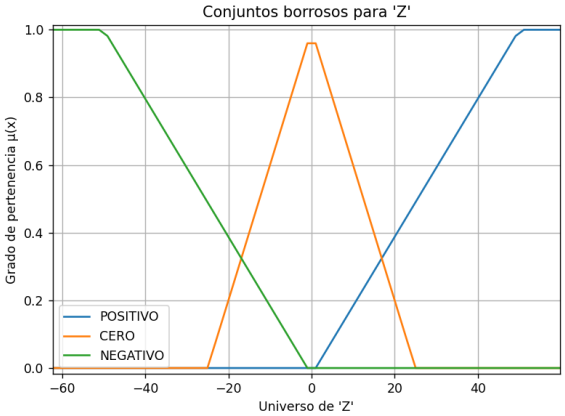
- Coordina todo el proceso
- Obtiene entradas del Environment
- Ejecuta inferencia con FIS
- Aplica defuzzificación
- Retorna acción de control (apertura ventana)

Conjuntos Difusos: Variable 'Hora'



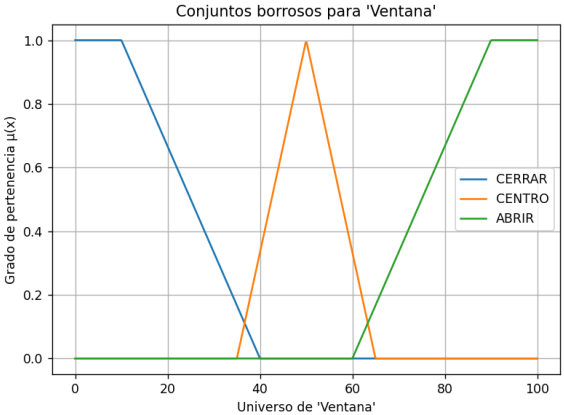
- Variable sin difusión entre conjuntos
- Conjuntos trapezoidales bien delimitados
- "DIA": 8:00-20:00
- "NOCHE": 0:00-8:00
- "NOCHE2": 20:00-24:00
- Transición abrupta entre día y noche

Conjuntos Difusos: Variable 'Z'



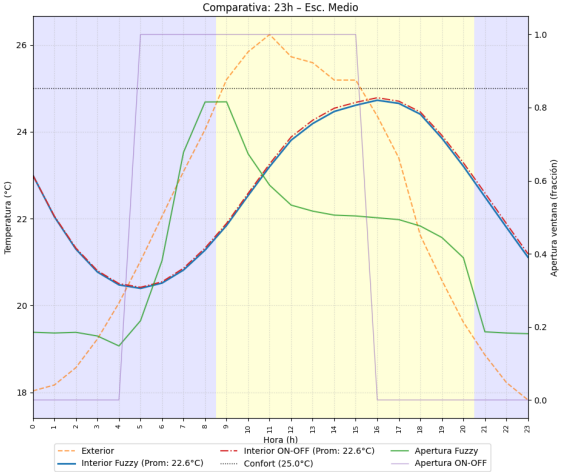
- Variable con difusión entre conjuntos
- "POSITIVO": $z > 0$ (cerrar ventana)
- "ZERO": $z \approx 0$ (posición intermedia)
- "NEGATIVO": $z < 0$ (abrir ventana)
- Transiciones suaves entre conjuntos
- Solapamiento parcial entre conjuntos

Conjuntos Difusos: Variable 'Ventana'

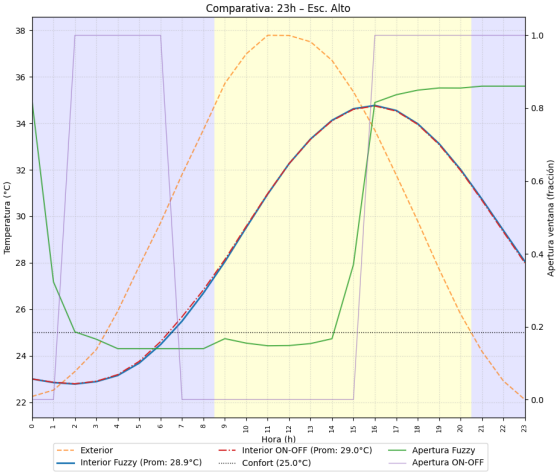


- Variable de salida
- "CERRAR": 0-40 % (ventana casi cerrada)
- "CENTRO": 35-65 % (posición intermedia)
- "ABRIR": 60-100 % (ventana abierta)
- Solapamiento entre conjuntos para transiciones suaves
- COG determina el valor final de apertura

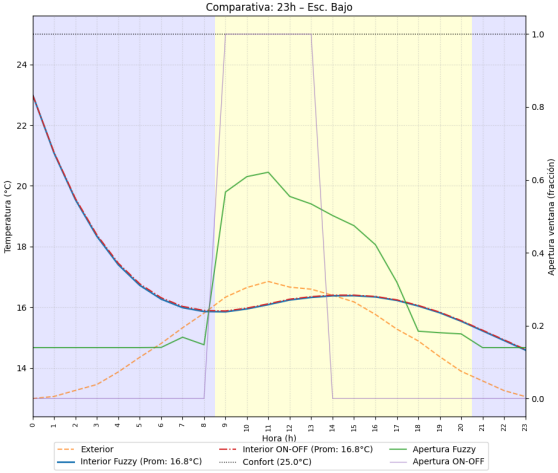
Simulación - Serie media - COG



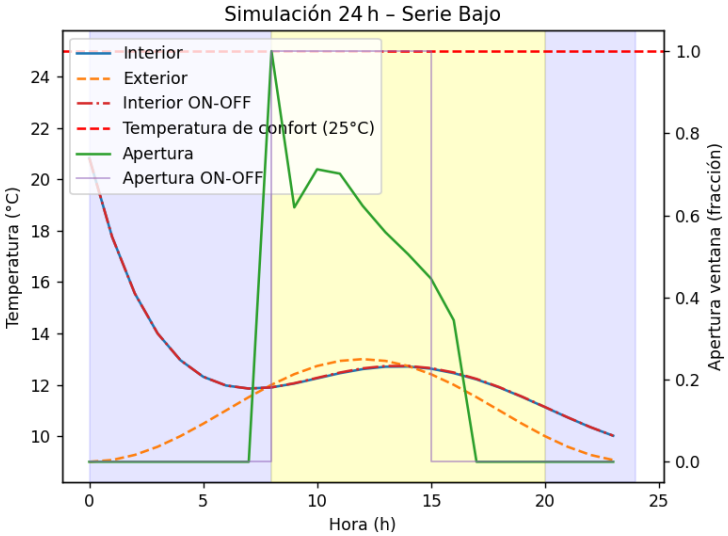
Simulación - Serie alta - COG



Simulación - Serie baja - COG

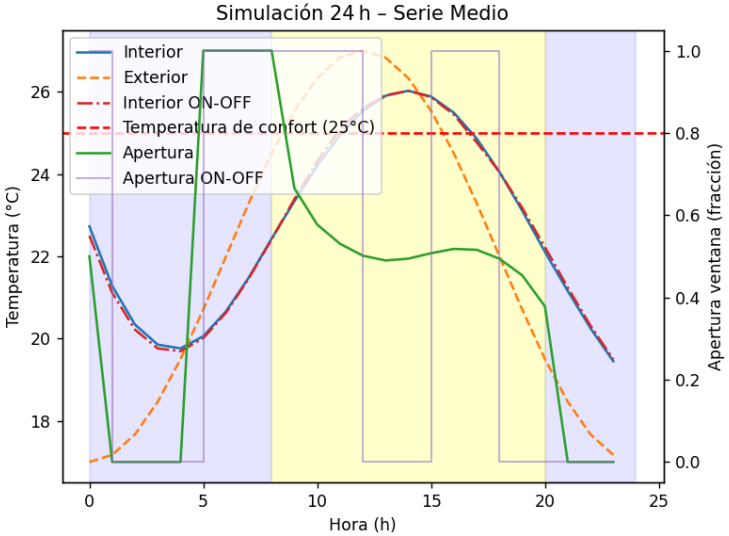


Simulación - Serie Baja (Frío) - COG++



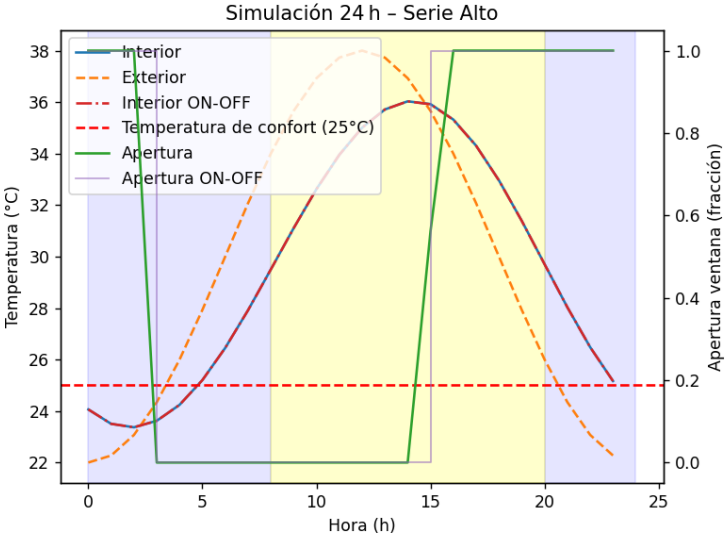
- Serie exterior siempre menor que 15°C
- Durante el día:
  - Ventana se abre para aprovechar el calor máximo exterior
- Durante la noche:
  - $T_{predicha}$ BAJA
  - Consigna $v_0 = 50C$
  - Ventana cerrada para retener calor

Simulación - Serie Media (Templado) - COG++



- Serie exterior alrededor de 25°C
- Durante el día:
  - Ventana se abre/cierra ajustando temperatura
- Durante la noche:
  - $T_{predicha}$ CERO
  - Consigna $v_0 = 25C$
  - Comportamiento equilibrado
- Suavidad del control difuso vs. On-Off

Simulación - Serie Alta (Calor) - COG++



- Serie exterior siempre ¿ 25°C
- Durante el día:
  - Ventana cerrada cuando exterior ¿ interior
  - Se abre cuando exterior desciende
- Durante la noche:
  - $T_{predicha}$ ALTA
  - Consigna $v_0 = 5C$
  - Ventana abierta para enfriamiento anticipado

Comparación con Control On-Off

- El control difuso demuestra ventajas significativas:
  - Transiciones más suaves (sin oscilaciones bruscas)
  - Menor desgaste en el actuador (ventana)
  - Mejor mantenimiento de la temperatura cercana al confort
  - Anticipación a las condiciones del día siguiente
- La difusión en la variable Z permite respuestas proporcionales
- La estrategia de adaptación nocturna mejora el rendimiento global

Nota importante

El sistema implementa difusión únicamente en la variable Z, mientras que las variables Hora y Temperatura predicha se manejan de forma discreta.

Conclusiones

- El sistema de control difuso demuestra ser efectivo para:
  - Adaptarse a diferentes condiciones climáticas
  - Mantener la temperatura cercana al confort (25°C)
  - Anticipar condiciones térmicas futuras
  - Sistemas intrínsecamente no lineales, como el caso estudiado.
- La arquitectura modular orientada a objetos permite:
  - Fácil extensibilidad
  - Reconfiguración de reglas y parámetros
  - Reutilización para otros sistemas de control
- Posibles mejoras:
  - Añadir más variables de entrada (humedad, viento)
  - Implementar aprendizaje para ajuste automático de reglas
  - Optimizar parámetros con algoritmos genéticos

Referencias

- S. Russel y P. Norvig, “Artificial Intelligence: A Modern Approach”
- T. J. Ross, “Fuzzy Logic with Engineering Applications”
- E.H. Mamdani, “Application of fuzzy control algorithms for control of simple dynamic plant”, Proc. IEE, 121 (12) (1974)
- J. M. Mendel, “Fuzzy Logic Systems for Engineering: A Tutorial”, Proceedings of the IEEE, 83(3), 345-377, 1995
- M. Marchetta, “Unidad 2 - Lógica Difusa”, Material de la cátedra, UNCUYO