Practica 5. Lógica de primer orden

La lógica de primer orden se refiere a una lógica formal que permite cuantificar sobre individuos en un dominio, como en los ejemplos de "para todo" (∀) y "existe" (∃).

Si bien MATLAB no tiene una biblioteca específica para lógica de primer orden, se pueden implementar conceptos utilizando matrices, funciones lógicas y programación simbólica.

Ejemplo 1: Verificación de cuántos individuos cumplen una propiedad

En este ejemplo, queremos verificar si **todos los elementos** de un conjunto cumplen una propiedad (un predicado `P(x)`), o si existe al menos uno que la cumple.

```
% Definir un conjunto de individuos
X = [1, 2, 3, 4, 5, 6];

% Definir el predicado P(x): 'x es un número par'
P = @(x) mod(x, 2) == 0;

% Cuantificador universal: verificar si todos los elementos cumplen P(x)
todos_cumplen = all(arrayfun(P, X));

% Cuantificador existencial: verificar si existe al menos un elemento que cumple
P(x)
existe_al_menos_uno = any(arrayfun(P, X));

fprintf('Todos cumplen P(x): %d\n', todos_cumplen);
fprintf('Existe al menos uno que cumple P(x): %d\n', existe_al_menos_uno);
```

Ejemplo 2: Verificar si una regla lógica es válida en un dominio

Supongamos que queremos verificar si una implicación lógica se cumple para un dominio dado. Por ejemplo, queremos verificar si para todos los elementos `x` en el conjunto `X`, si `x` es par, entonces `x` es mayor que 1.

```
% Definir el conjunto de individuos
X = [0, 1, 2, 3, 4, 5];

% Definir los predicados
P = @(x) mod(x, 2) == 0;  % P(x): 'x es par'
Q = @(x) x > 1;           % Q(x): 'x es mayor que 1'

% Cuantificador universal para la implicación P(x) -> Q(x)
todos_cumplen_implicacion = all(arrayfun(@(x) ~P(x) || Q(x), X));

fprintf('Todos los elementos cumplen P(x) -> Q(x): %d\n',
todos_cumplen_implicacion);
```

Ejemplo 3: Lógica cuantificacional con programación simbólica

MATLAB también permite trabajar con álgebra simbólica, que es útil para representar proposiciones lógicas más complejas.

```
% Usar el toolbox simbólico de MATLAB
syms x

% Definir un predicado simbólico
P = (mod(x, 2) == 0); % P(x): 'x es par'
Q = (x > 1);          % Q(x): 'x es mayor que 1'

% Verificar la implicación P(x) -> Q(x) simbólicamente
implicacion = implies(P, Q);
disp(implicacion);
```

Ejemplo 4: Cuantificadores en dominios más complejos

Supongamos que tenemos un conjunto de pares de números y queremos verificar si existe un par de números tal que el primer número es mayor que el segundo.

```
% Definir un conjunto de pares de números
X = [1 2; 3 1; 5 6; 4 4];

% Definir el predicado P(x, y): 'x > y'
P = @(x, y) x > y;

% Cuantificador existencial: verificar si existe un par que cumple P(x, y)
existe_al_menos_un_par = any(arrayfun(@(i) P(X(i, 1), X(i, 2)), 1:size(X, 1)));

fprintf('Existe al menos un par que cumple P(x, y): %d\n',
existe_al_menos_un_par);
```

Explicación:

1. **`arrayfun`**: Aplica una función a cada elemento del arreglo.
2. **`all`**: Verifica si todos los elementos de un vector son verdaderos.
3. **`any`**: Verifica si al menos uno de los elementos de un vector es verdadero.
4. **`implies`**: Simula una implicación lógica (esto puede definirse a mano en MATLAB si no tienes el paquete simbólico).

Estos ejemplos cubren algunas operaciones básicas que puedes realizar en MATLAB utilizando la lógica de primer orden.

Problema:

Una agencia de control ambiental quiere determinar si la calidad del agua en diferentes fuentes cumple con los estándares de seguridad. Los estándares especifican que:

- El nivel de pH debe estar entre 6.5 y 8.5.
- La concentración de nitratos debe ser menor a 50 mg/L.

Se han recopilado datos de diferentes fuentes de agua en la ciudad, con los niveles de pH y nitratos correspondientes. El objetivo es determinar si **todas las fuentes** cumplen los estándares o si existe alguna fuente que **no cumpla** con al menos uno de los criterios.

Lógica Formal:

Definimos el siguiente sistema de lógica de primer orden:

- Sea x el conjunto de fuentes de agua.
- P(x): "El nivel de pH de la fuente x está entre 6.5 y 8.5".
- Q(x): "La concentración de nitratos de la fuente x es menor a 50 mg/L".
- El estándar de seguridad para una fuente de agua x es S(x), donde $S(x)=P(x) \wedge Q(x) S(x)$ ("El nivel de pH y la concentración de nitratos cumplen los estándares").

El objetivo es verificar si $\forall x\, S(x)$, es decir, si todas las fuentes cumplen con los estándares de seguridad, o si $\exists x\, \neg S(x)$, es decir, si existe al menos una fuente que no los cumple.

```
% Datos de las fuentes de agua: [pH, nitratos (mg/L)]
fuentes = [
    7.2, 30;  % Fuente 1: pH = 7.2, nitratos = 30 mg/L
    8.0, 45;  % Fuente 2: pH = 8.0, nitratos = 45 mg/L
    6.3, 40;  % Fuente 3: pH = 6.3, nitratos = 40 mg/L (pH fuera de rango)
    7.8, 55;  % Fuente 4: pH = 7.8, nitratos = 55 mg/L (nitratos fuera de rango)
    6.9, 20   % Fuente 5: pH = 6.9, nitratos = 20 mg/L
];

% Definir el predicado P(x): pH entre 6.5 y 8.5
P = @(ph) (ph >= 6.5) & (ph <= 8.5);

% Definir el predicado Q(x): nitratos < 50 mg/L
Q = @(nitratos) nitratos < 50;

% Verificar si cada fuente cumple los estándares
S = @(ph, nitratos) P(ph) & Q(nitratos);

% Aplicar la función a todas las fuentes
```

```
cumplimiento = arrayfun(@(i) S(fuentes(i, 1), fuentes(i, 2)), 1:size(fuentes,
1));

% Verificar si todas las fuentes cumplen con los estándares
todas_cumplen = all(cumplimiento);
existe_al_menos_una_no_cumple = any(~cumplimiento);

% Resultados
if todas_cumplen
    fprintf('Todas las fuentes de agua cumplen con los estándares de
calidad.\n');
else
    fprintf('Al menos una fuente de agua no cumple con los estándares de
calidad.\n');
    fprintf('Fuentes que no cumplen:\n');
    disp(find(~cumplimiento));  % Muestra el índice de las fuentes que no cumplen
end
```

Contexto teórico:

Este problema puede modelarse mediante **cuantificadores lógicos** que evalúan predicados asociados a propiedades fisicoquímicas del agua. El uso de la lógica de primer orden permite la expresión formal de las condiciones de seguridad, representadas por los predicados P(x) y Q(x). En este caso, la solución se basa en un análisis exhaustivo del conjunto de datos, utilizando cuantificadores universales ∀x y existenciales ∃x, lo que permite verificar si todas las fuentes cumplen con los estándares o si alguna de ellas no lo hace.

Solución:

- **Predicado P(x)**: Evalúa si el nivel de pH de cada fuente está dentro del rango permitido (entre 6.5 y 8.5).
- **Predicado Q(x)**: Evalúa si la concentración de nitratos es inferior al límite de 50 mg/L.
- **Cuantificador universal ∀x**: Se verifica si todas las fuentes cumplen simultáneamente ambos criterios.
- **Cuantificador existencial ∃x**: Determina si hay al menos una fuente que no cumpla alguna de las condiciones de seguridad.

El programa en MATLAB evalúa estos predicados para cada fuente de agua de la ciudad, y si se encuentra alguna que no cumpla con los estándares, proporciona un reporte con las fuentes específicas que incumplen.

Resultados:

En el ejemplo proporcionado, dos de las cinco fuentes no cumplen con los estándares:

- La **Fuente 3** tiene un pH inferior a 6.5.
- La **Fuente 4** tiene una concentración de nitratos superior a 50 mg/L.

El programa señala estas fuentes, facilitando la toma de decisiones para intervenir en el sistema de distribución de agua y asegurar la calidad.

Conclusión:

Este ejemplo demuestra cómo la lógica de primer orden puede aplicarse para resolver problemas relacionados con la salud pública y la calidad del agua, utilizando MATLAB para modelar y analizar los datos cuantitativos. Este enfoque es útil en múltiples áreas de investigación, tales como la gestión ambiental, la ingeniería civil y la bioestadística, donde la verificación de condiciones y normas es fundamental para la toma de decisiones informadas.

Este tipo de análisis puede extenderse a otros criterios y conjuntos de datos, facilitando una evaluación más compleja de la calidad del agua o cualquier otro recurso ambiental.

Ejercicio propuesto:

Una empresa de distribución de mercancías tiene varios centros de distribución (CDs) y clientes ubicados en diferentes ciudades. El objetivo es determinar si es posible entregar todos los pedidos en un tiempo máximo determinado (por ejemplo, 48 horas) desde algún centro de distribución cercano. Si no es posible, se deben identificar los clientes que no pueden ser atendidos dentro del límite de tiempo.

Planteamiento:

Lógica Formal:

Definimos el siguiente sistema de lógica de primer orden:

- Sea x el conjunto de **clientes**.
- Sea y el conjunto de **centros de distribución (CDs)**.
- D(x,y)D(x, y)D(x,y): "La distancia entre el cliente x y el centro y".
- T(x,y)T(x, y)T(x,y): "El tiempo estimado de entrega desde el centro y al cliente x".
- La entrega desde un centro y a un cliente x es posible si $T(x,y) \leq t_{max} T(x, y)$, donde $t_{max}=48$ horas es el límite máximo permitido.

Queremos verificar si **todos los clientes** pueden ser atendidos desde **algún centro** dentro del tiempo permitido, es decir, $\forall x \, \exists y \, T(x,y) \leq t_{max}$.