



# Problema de los interruptores y las bombillas

## Proyecto final

lógica para ciencias de la computación

- Joseph Doqueresama
- Ivan Moreno

# Situación a representar

En una habitación hay tres interruptores y tres bombillas, pero no está claro cuál interruptor controla cada bombilla. Solo puedes ingresar a la habitación una vez. El objetivo es determinar qué interruptor está asociado con qué bombilla utilizando la información observada después de una sola visita a la habitación.

## Información disponible

1. **Interruptores:** Hay tres interruptores en la habitación, numerados del 1 al 3.
2. **Bombillas:** También hay tres bombillas, numeradas del 1 al 3.
3. **Estado inicial:** Al comienzo, todos los interruptores y bombillas están apagados.
4. **Acciones permitidas:** Solo puedes realizar dos acciones:
  - a. Encender uno de los interruptores.
  - b. Ingresar a la habitación.

# Situación a representar

## Restricciones

1. **Una sola visita:** Solo se permite ingresar a la habitación una vez.
2. **Unicidad de la iluminación:** Cuando se enciende un interruptor, solo una bombilla se enciende y las otras dos permanecen apagadas.
3. **Observación de resultados:** Después de encender un interruptor y entrar en la habitación, solo puedes observar el estado de las bombillas.

Con estas restricciones, aseguramos que cada acción tenga un efecto único y claro en el estado de las bombillas, lo que nos permite deducir correctamente la asociación entre los interruptores y las bombillas.

# Situación a representar

## Estrategia de resolución

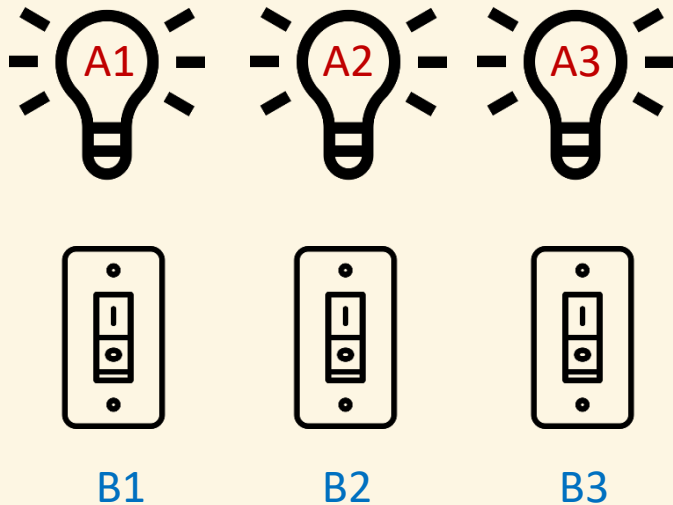
1. Enciende uno de los interruptores y deja que las bombillas permanezcan encendidas durante un tiempo corto.
2. Apaga el interruptor previamente encendido y enciende otro interruptor.
3. Ingresa a la habitación y observa el estado de las bombillas.
4. Utiliza las observaciones para deducir qué interruptor controla cada bombilla.

Encendemos el interruptor 1 y dejamos las bombillas encendidas brevemente. Luego, apagamos el interruptor 1 y encendemos el interruptor 2. Al entrar en la habitación, si identificamos la bombilla como encendida estará relacionada con el interruptor 2, si la bombilla está apagada pero caliente con el interruptor 1 y si la bombilla está apagada y fría con el interruptor 3.

Esta estrategia nos permite determinar qué interruptor controla cada bombilla con una sola visita.

# Representación en lógica proposicional

Considere tres (3) interruptores y tres (3) bombillas así:




Para esta situación, vamos a representar cada elemento dentro del problema con una letra:

**A1**, **A2** y **A3** hace referencia a las bombillas

**B1**, **B2** y **B3** hace referencia a los interruptores

# Letras Proposicionales

Para representar la situación en lógica, definiremos también cada una de las letras proposicionales:

- 
- **B1**: La bombilla 1 está encendida.
  - **B2**: La bombilla 2 está encendida.
  - **B3**: La bombilla 3 está encendida.
  - **I1**: El interruptor 1 está encendido.
  - **I2**: El interruptor 2 está encendido.
  - **I3**: El interruptor 3 está encendido.

# Reglas

De acuerdo con el planteamiento del problema podemos enunciar las siguientes reglas:

Regla 1: Si una bombilla está encendida, su interruptor también debe estar encendido.

**Regla completa:**

$$\bigwedge_{b,i} (Pertenece(b,i) \leftrightarrow (\bigvee_e bombilla(b,e) \wedge interruptor(i,e)))$$

Regla 2: todo bombillo le pertenece algún interruptor.

**Regla completa:**

$$\bigwedge_b (Estado(b) \leftrightarrow (\bigvee_i pertenece(b,i) \wedge Estado(i)))$$

# Reglas

De acuerdo con el planteamiento del problema podemos enunciar las siguientes reglas:

Regla 3: Todo interruptor pertenece a algún bombillo

**Regla completa:**

$$\bigwedge_i (\text{Estado}(i) \leftrightarrow (\bigvee_b \text{pertenece}(b, i) \wedge \text{Estado}(b)))$$

Regla 4: Un bombillo no puede tener más de un interruptor.

**Regla completa:**

$$\bigwedge_{b,i} (\text{Pertenece}(b, i) \rightarrow \neg (\bigvee_{i' \neq i} \text{pertenece}(b, i') )$$



# Reglas

De acuerdo con el planteamiento del problema podemos enunciar las siguientes reglas:

Regla 5: Un interruptor no pertenece a más de un bombillo.

**Regla completa:**

$$\bigwedge_{i,b} (\text{Pertenece}(b,i) \rightarrow \neg (\bigvee_{b' \neq b} \text{pertenece}(b',i)))$$

# Solución

Se plantea una situación inicial donde al ingresar a la habitación tenemos que el bombillo 2 se encuentra encendido, y nosotros encendimos y luego apagamos el interruptor 1

- Al inicializar el problema, van a empezar a correr todas las reglas dentro de los 3 descriptores definidos para el problema; así el sistema logra identificar cada estado y lo relaciona al problema inicial:

```
# Instanciando descriptores para clase
self.Bombillos = Descriptor([3,3])
self.Pertenece = Descriptor([3,3], chrInit = 256 + 9)
self.Interruptor = Descriptor([3,3], chrInit = 256 + 18)
# Modificando método describir para que escriba cada descriptor
self.Bombillos.escribir = MethodType(escribirBombillas,self.Bombillos)
self.Pertenece.escribir = MethodType(escribirPertenenencia,self.Pertenece)
self.Interruptor.escribir = MethodType(escribirInterruptor,self.Interruptor)
```

# Solución

Después, al obtener la formula en formato `in_order_to_tree()` pasamos dicha formula al algoritmo `tseitin()` para transformar esa formula en su forma clausal, clausulas que luego almacenamos en un diccionario y filtramos para poder eliminar todas las letras extra que son generadas por el algoritmo de `tseitin()`

```
tseitinn = tseitin(Ytoria(problemaa.regla)) # pasar todo el problema a tseitin (obtener el problema de forma clausal )
tseitinn
```

```
result, dicc = dpll(tseitinn, {}) # paso el seitin al ddpl y luego guardo las representaciones en un ciccionario vacio (V o F)
```

```
< for k in dicc: # recorre el diccionario
< |   if ord(k) < 256 + 27 and dicc[k]: # filtra (para sacar las letras extra se tseitin)
|   print(problemaa.escribirFormula(k), dicc[k]) #imprimir el resultado (solo lo verdadero)
```

# Solución

Una vez imprimimos el resultado satisficible del algoritmo tseitin, filtramos las letras extras y solo imprimimos los estados verdaderos del diccionario; llegamos a que el algoritmo logra llegar al resultado esperado y a travez del problema inicial; logra inferir el resto de estados y relaciones; haciendo uso de la estrategia de solución planteada

```
for k in dicc:# recorre el diccionario
|   if ord(k) < 256 + 27 and dicc[k]:# filtra (para sacar las letras extra se tseitin)
|       print(problemaa.escribirFormula(k), dicc[k]) #imprimir el resultado (solo lo verdadero)
```

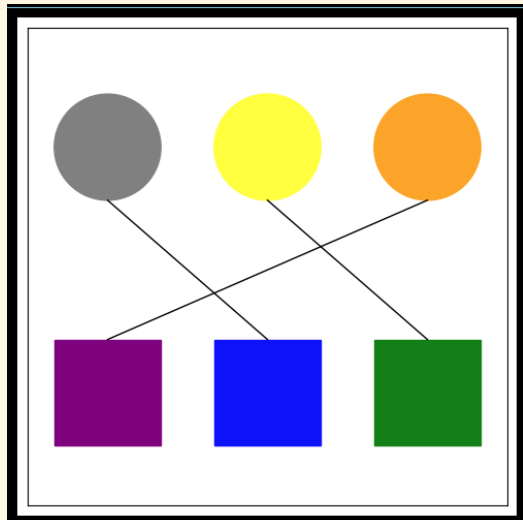
```
El bomillo 2 está Encendido True
El interruptor 3 está Encendido True
El Interruptor 3 pertenece a bombillo 2 True
El Interruptor 1 pertenece a bombillo 3 True
El Interruptor 2 pertenece a bombillo 1 True
El bomillo 3 está Caliente True
El bomillo 1 está Apagado True
El interruptor 2 está Apagado True
El interruptor 1 está Encendido y luego apagado True
```

Podemos ver además, como desde esta estrategia logra deducir que interruptor esta relacionado con que bombillo.

# Conclusiones

Al aplicar esta estrategia, el algoritmo desarrollado demuestra su capacidad para resolver el problema de manera satisfactoria, incluso cuando se enfrenta a diferentes situaciones iniciales, como se ilustra con el ejemplo proporcionado donde se presenta un escenario específico y se muestra cómo el algoritmo lo resuelve de manera precisa.

En conclusión, la estrategia de resolución propuesta no solo permite determinar con precisión qué interruptor controla cada bombilla, sino que también demuestra la utilidad de la lógica proposicional y los algoritmos adecuados para abordar problemas complejos de este tipo con eficacia y eficiencia.



colores bombillas:

1. **Amarillo** -> encendido
2. **Gris** -> apagado
3. **Naranja** -> caliente (se encendio y luego se apago)

colores interruptores:

1. **azul** -> apagado
2. **verde** -> encendido
3. **morado** -> encendido y luego apagado

# Gracias

