Lógica Proposicional en la Programación

Materia: Algoritmos y Programación

Prof. D.Sc. BARSEKH-ONJI Aboud
Facultad de Ingeniería
Universidad Anáhuac México

7 de diciembre de 2025

Agenda

Lógica Proposicional en la Programación

Más Allá de las Decisiones Simples

¿Qué sucede cuando una decisión depende de múltiples condiciones?

'Si un estudiante tiene un promedio > 90 **Y** no tiene faltas, **ENTONCES** obtiene una beca'.

Para manejar estas situaciones, la programación se apoya en los principios de la **lógica proposicional** o **álgebra de Boole**.

Lógica Proposicional en la Programación

Valores Booleanos y Operadores Lógicos

Esta rama de la lógica trabaja con valores que solo pueden ser **Verdadero ('True')** o **Falso ('False')**. En programación, estos valores se conocen como **booleanos**. Toda decisión en un programa se reduce a evaluar si una expresión booleana es verdadera o falsa.

Operador Lógico: AND (Conjunción)

AND (Y)

Evalúa si **ambas** proposiciones son verdaderas. El resultado es `Verdadero` únicamente si la Proposición A **y** la Proposición B son verdaderas.

Tabla de Verdad

A	B	A AND B
V	V	**V**
V	F	F
F	V	F
F	F	F

Examples

Algorithm 1 ¿Puedo conducir el vehículo?

```
1: if (edad ¿= 18) AND (tieneLicencia = VERDADERO) then
2:     ESCRIBIR 'Puede conducir.'
3: end if
```

Operador Lógico: OR (Disyunción)

OR (O)

Evalúa si **al menos una** de las proposiciones es verdadera. Solo es `Falso` si ambas proposiciones son falsas.

Tabla de Verdad

A	B	A OR B
V	V	**V**
V	F	**V**
F	V	**V**
F	F	F

Examples

Algorithm 2 ¿Hoy es un día libre?

1: **if** (esFeriado = **VERDADERO**) **OR** (dia = 'Sábado') **then**
2: **ESCRIBIR** 'Hoy no se trabaja.'
3: **end if**

Operadores Lógicos: NOT y XOR

NOT (Negación)

Es un operador unario: actúa sobre una sola proposición e **invierte** su valor de verdad.

A	NOT A
V	F
F	V

Algorithm 3 ¿Salir a caminar?

```
1: if NOT (estaLloviendo) then
2:     ESCRIBIR '¡Sal a caminar!'
3: end if
```

Operadores Lógicos: NOT y XOR

XOR (Disyunción Exclusiva)

Devuelve `Verdadero` solo si las dos proposiciones tienen valores de verdad **diferentes**.

A	B	A XOR B
V	V	F
V	F	**V**
F	V	**V**
F	F	F

Algorithm 4 Modo de Visualización

```
1: if (modoNocturno) XOR (modoDiurno) then
2:     ESCRIBIR 'Estado válido.'
3: else
4:     ESCRIBIR 'Error de estado.'
5: end if
```

Tablas de Verdad

¿Qué es una Tabla de Verdad?

Es una herramienta matemática que desglosa una expresión lógica compleja en todas sus posibles combinaciones de valores de verdad ('Verdadero' / 'Falso'), mostrando el resultado final para cada caso.

Tablas de Verdad

Utilidad en Programación

Nos permiten:

- **Verificar la lógica:** Asegurarnos de que una condición compleja se comporta como esperamos.
- **Simplificar expresiones:** Encontrar una forma más simple y eficiente de escribir la misma condición.
- **Depurar errores:** Encontrar la causa de un comportamiento inesperado en una estructura condicional.

Tamaño de la Tabla

El número de filas en una tabla de verdad es $\mathbf{2^n}$, donde n es el número de variables distintas en la expresión.

Construcción de Tablas de Verdad

Ejemplo 1: (A OR B) AND (NOT A)

A	B	NOT A	A OR B	Resultado
V	V	F	V	**F**
V	F	F	V	**F**
F	V	V	V	**V**
F	F	V	F	**F**

Ejemplo 2: (A AND B) OR C

A	B	C	A AND B	Resultado
V	V	V	V	**V**
V	V	F	V	**V**
V	F	V	F	**V**
V	F	F	F	**F**
F	V	V	F	**V**
F	V	F	F	**F**
F	F	V	F	**V**
F	F	F	F	**F**

Tautologías y Contradicciones

Tautología

Es una expresión que es **siempre verdadera**, sin importar los valores de sus componentes.

A OR (NOT A)

A	NOT A	Resultado
V	F	**V**
F	V	**V**

Contradicción

Es una expresión que es **siempre falsa**, sin importar los valores de sus componentes.

A AND (NOT A)

A	NOT A	Resultado
V	F	**F**
F	V	**F**

Teoremas de De Morgan

¿Qué son?

Dos teoremas formulados por Augustus De Morgan que proporcionan un método poderoso para transformar y **simplificar expresiones lógicas**, especialmente aquellas que involucran negaciones de operaciones AND y OR.

Teorema 1: Negación de un AND

La negación de una conjunción (AND) es la disyunción (OR) de las negaciones.

```
NOT (A AND B)
      ⇕
(NOT A) OR (NOT B)
```

Teorema 2: Negación de un OR

La negación de una disyunción (OR) es la conjunción (AND) de las negaciones.

```
NOT (A OR B)
      ⇕
(NOT A) AND (NOT B)
```

Aplicación Práctica en Programación

1. Simplificar Condiciones Complejas

Las condiciones negadas son difíciles de leer. Aplicar las leyes de De Morgan las hace más intuitivas.

Condición Confusa:

```
SI NO ((esCliente=FALSO) O (deuda>0)) ENTONCES...
```

Condición Simplificada (equivalente):

```
SI (esCliente=VERDADERO) Y (deuda<=0) ENTONCES...
```

2. Formular Condiciones de Bucle

Son muy útiles en bucles `MIENTRAS`. A menudo es más fácil definir la condición para **terminar** y luego negarla para obtener la condición de **continuación**.

Condición para terminar un juego:

```
(vidas = 0) O (tiempoAgotado)
```

Condición para continuar el bucle (negada y simplificada):

```
MIENTRAS (vidas > 0) Y (NO tiempoAgotado) HACER...
```

Homework: Validador de Acceso para Montaña Rusa

Objetivo

Escribir un programa en Python que determine si una persona puede subir a una montaña rusa, basándose en un conjunto de reglas de seguridad que combinan múltiples condiciones lógicas (‘and‘, ‘or‘, ‘not‘).

Homework: Validador de Acceso para Montaña Rusa

Requisitos del Programa

1. **Recopilar Datos del Usuario:** El programa debe solicitar la siguiente información:
   - La edad de la persona (como un número entero).
   - La altura de la persona en metros (puede ser un número con decimales).
   - Si trae puesto un pase especial 'V.I.P.' (responder 'Sí' o 'No').
2. **Implementar las Reglas de Acceso:** Una persona puede subir **SOLAMENTE SI** se cumple la siguiente condición principal:
   - La persona mide más de 1.40 metros **Y** su edad está entre 10 y 60 años (inclusive).

   O BIEN, si la regla anterior no se cumple, hay una excepción:
   - La persona tiene un pase 'V.I.P.' **Y NO** es menor de 7 años.
3. **Mostrar el Resultado:**
   - Si la persona cumple las condiciones, el programa debe imprimir: `Acceso Concedido. ¡Disfrute la atracción!`
   - Si no las cumple, debe imprimir: `Acceso Denegado. No cumple con los requisitos de seguridad.`

Homework: Validador de Acceso para Montaña Rusa

Pistas y Consejos

- Recuerda convertir la edad a `int()` y la altura a `float()` después de pedirlas.
- Para la pregunta del pase V.I.P., puedes convertir la respuesta 'Sí'/'No' a un booleano. Una forma fácil es: `tiene_pase = input('...').lower() == 'sí'`.
- ¡La clave del problema está en construir una única y gran condición `if` que represente todas las reglas usando `and`, `or` y paréntesis `()` para agrupar la lógica correctamente!