Guía de Estudio en Lógica y Algoritmos
Curso Propedéutico de la
Licenciatura en Inteligencia Artificial

Jorge Hermosillo Valadez, Bruno Lara Guzmán
Departamento de Computación
Centro de Investigación en Ciencias
Universidad Autónoma del Estado de Morelos (UAEM)

8 de julio de 2022

Índice general

Prefacio

Esta obra es una guía de estudio para los temas de Lógica y Algoritmos del curso propedéutico de la Licenciatura en Inteligencia Artificial, del Instituto de Ciencias Básicas y Aplicadas, de la Universidad Autónoma del Estado de Morelos (UAEM).

Se abordan temas básicos sobre el pensamiento lógico y el diseño básico de algoritmos. La Inteligencia Artificial ancla parte de sus raíces en las ciencias de la computación, donde el pensamiento lógico y el diseño de algoritmos son temas sustantivos. En esta guía se introduce al estudiante al mundo del razonamiento lógico y la resolución de problemas basados en la lógica proposicional y el diseño de algoritmos sencillos.

Capítulo 1

Introducción a la Computación

Jorge Hermosillo Valadez

1.1. ENIAC: nace la computación moderna

Existe mucha información publicada acerca de la historia de las computadoras, por lo que no haremos aquí un recuento detallado de su evolución. Sin embargo, existe un hito en este proceso que vale la pena recordar por el impacto que tuvo en el diseño de las computadoras, produciendo un cambio que revolucionó la concepción de la computación, habilitando la posibilidad de crear un sinnúmero de soluciones a problemas de diversa índole. Se trata de la primera computadora electrónica, que funcionaba con tubos al vacío, llamada Electronic Numerical Integrator And Calculator (ENIAC).

La ENIAC se construyó en la Universidad de Pennsylvania en 1947, por un equipo de diseño encabezado por los ingenieros John Mauchly y John Eckert. En el diseño de la ENIAC fueron incluidas nuevas técnicas de la electrónica que permitían minimizar el uso de partes mecánicas. Esto trajo como consecuencia un incremento significativo en la velocidad de procesamiento: podía efectuar 5,000 sumas o 500 multiplicaciones en un segundo y permitía el uso de aplicaciones científicas en astronomía, meteorología, etc. Esta máquina tenía más de 18,000 tubos de vacío, consumía 200 KW de energía eléctrica y requería todo un sistema de aire acondicionado. El proyecto, auspiciado por el departamento de Defensa de los Estados Unidos, culminó dos años después, cuando se integró a ese equipo el ingeniero y matemático húngaro-estadounidense John von Neumann (1903 - 1957).

Durante el desarrollo del proyecto ENIAC, von Neumann propuso mejoras que posibilitaron el desarrollo de las computadoras actuales:

1. Utilizar un **sistema de numeración de base dos (binario)** en vez del sistema de-

cimal tradicional.

2. Hacer que las instrucciones de operación estén en la memoria, al igual que los datos. De esta forma, memoria y programa residirán en un mismo sitio.

Con estos cambios, la ENIAC revolucionó el mundo de las computadoras, y con ellos, nace la era de la computación moderna.

1.2. Números binarios

1.2.1. Noción de sistema posicional

Nuestro sistema de números decimales, funciona como un **sistema posicional**. En un sistema posicional cada dígito posee un valor determinado por **la base** del sistema y la posición relativa del dígito.

Definición 1.1. La **base de un sistema numérico** es el *número de símbolos (guarismos) utilizados para representar cualquier número del sistema.*

Ejemplo 1.1

Por ejemplo,
*El **sistema decimal** está formado por **10 símbolos** $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ por lo tanto su base es 10.*
*El **sistema binario** está formado por **2 símbolos** $\{0, 1\}$ por lo tanto su base es 2.*
*El **sistema hexadecimal** está formado por **16 símbolos** $\{0\ al\ 9\}$ y $\{A, B, C, D, E, F\}$, que representan los números 10 al 15 respectivamente, por lo tanto su base es 16.*

La posición de cada símbolo le otorga un peso que es indicativo del valor del símbolo. En el sistema decimal, a estos pesos les llamamos: unidades, decenas, centenas, etc.

Ejemplo 1.2

Por ejemplo, en nuestro sistema numérico, cuando observamos la cadena de números: 235, decimos que hay 5 unidades (5x1), 3 decenas (3x10), y 2 centenas (2x100), por lo que implícitamente, el valor de este número es: 200+30+5 = 235. Matemáticamente, escribimos:

$$235 = 2 \times (10)^2 + 3 \times (10)^1 + 5 \times (10)^0$$

Para usar un sistema de notación posicional, es importante contar con el cero. De esta forma, el guarismo 0 permite distinguir entre 11, 101 y 1001 sin tener que agregar símbolos

adicionales, gracias a la noción de base. Así, en nuestro sistema decimal, todos los números se pueden expresar con sólo diez guarismos, del 1 al 9 más el 0. Nota que el número 10, es la concatenación del dígito 1 y el 0, que se logra cuando llegamos al 9 e incrementamos de 1 para obtener el 10.

Ejercicio Resuelto 1.1. Diga si el número es correcto dada la base que se proporciona.

1. base 2: 11011
2. base 3: 11011
3. base 3: 322
4. base 4: 333
5. base 8: $720A$
6. base 16: $7F0A$

SOLUCIÓN

1. base 2: 11011, correcto, todos los guarismos son válidos.
2. base 3: 11011, correcto, todos los guarismos son válidos.
3. base 3: 322, incorrecto, el guarismo 3 no existe en base 3.
4. base 4: 333, correcto, todos los guarismos son válidos.
5. base 8: $720A$, incorrecto, el guarismo A no existe en base 8.
6. base 16: $7F0A$, correcto, todos los guarismos son válidos.

1.2.2. Números binarios como unidades de información

Como hemos dicho, la computadora sólo es capaz de manipular dígitos binarios. Por lo tanto, cualquier tipo de información —números, letras, imágenes, etc.— está codificada en una computadora mediante 1s y 0s.

Para referirnos a cualquiera de estos dígitos binarios, se utiliza el término **'bit' (binary digit)**. Un bit permite entonces codificar un valor numérico de 0 o 1.

Definición 1.2. **Un byte** es una *secuencia ordenada de 8 bits* (Figura 1.1). Es la unidad básica de información en términos de procesamiento y almacenamiento computacional.

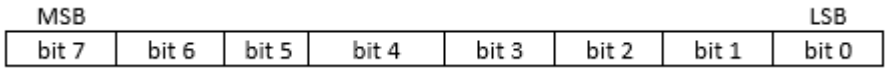


Figura 1.1: Típica organización de 1 Byte como una palabra de 8 bits.

Siendo el sistema binario un sistema posicional, cada bit en un byte tiene un peso que se incrementa de derecha a izquierda; de ahí los términos "Bit Menos Significativo" (LSB: Less Significant Bit) y "Bit Más Significativo" (MSB: Most Significant Bit).

1.2.3. Conversión binario-decimal

Al usar un **sistema binario**, las computadoras sólo pueden manipular los dígitos 0 y 1, así como un número finito de combinaciones de éstos.

Ejemplo 1.3

Por ejemplo, las siguientes cadenas binarias representan los números decimales 2, 3, 10 y 17:

decimal	*binario*
2	10
3	11
10	1010
17	1 0001

Para saber qué cantidad decimal representa una cadena binaria, podemos hacer una **conversión de la base binaria a la base decimal** de la siguiente forma.

Definición 1.3. Conversión binario-decimal. Dado un número binario de n bits $(d_{n-1}, d_{n-2}, \ldots, d_1, d_0)_2$, donde d_i representa el guarismo 0 o 1, su valor equivalente en decimal es.

$$N_{10} = d_{n-1} \times (2)^{n-1} + \cdots + d_2 \times (2)^2 + d_1 \times (2)^1 + d_0 \times (2)^0 \quad (1.1)$$

$$= \sum_{i=0}^{n-1} d_i \cdot (2)^i \quad (1.2)$$

Ejemplo 1.4

Retomemos los números binarios del Ejemplo 1.3 y hagamos la conversión a decimal.

binario	*decimal*
10	$1 \cdot (2)^1 + 0 \cdot (2)^0 = 2 + 0 = 2$
11	$1 \cdot (2)^1 + 1 \cdot (2)^0 = 2 + 1 = 3$
1010	$1 \cdot (2)^3 + 0 \cdot (2)^2 + 1 \cdot (2)^1 + 0 \cdot (2)^0 = 8 + 0 + 2 + 0 = 10$
1 0001	$1 \cdot (2)^4 + 1 \cdot (2)^0 = 16 + 1 = 17$

Observación 1.1. En general, podemos hacer una **conversión desde cualquier base a la base decimal** de la siguiente forma.

$$N_{10} = \sum_{i=0}^{n-1} d_i \cdot (b)^i \tag{1.3}$$

donde b es la base de origen, y d_i representa un guarismo válido en b.

Ejercicio Resuelto 1.2. Determine el valor decimal de los siguientes números, cada uno expresado en su base de origen.

1. $(11011)_2$
2. $(0111)_2$
3. $(322)_4$
4. $(1121)_3$

SOLUCIÓN

1. $(11011)_2 = 2^4 + 2^3 + 2^1 + 2^0 = 16 + 8 + 2 + 1 = 27$
2. $(0111)_2 = 2^2 + 2^1 + 2^0 = 4 + 2 + 1 = 7$
3. $(322)_4 = 3 \cdot 4^2 + 2 \cdot 4^1 + 2 \cdot 4^0 = 3(16) + 2(4) + 2(1) = 58$
4. $(1021)_3 = 1 \cdot 3^3 + 2 \cdot 3^1 + 1 \cdot 3^0 = 9 + 6 + 1 = 16$

1.2.4. Conversión entre bases potencia de 2

Existe una forma de convertir números binarios a la base **hexadecimal** (base 16=2^4), y viceversa, de manera muy sencilla y rápida.

Cada **dígito hexadecimal** equivale a un conjunto de **4 dígitos binarios**, tomados de derecha a izquierda. Veamos el siguiente ejemplo.

Ejemplo 1.5

Un número binario $1\ 1111 = 2^4 + 2^3 + 2^2 + 2^1 + 1 = 31$ *convertido a hexadecimal:*

$$\underbrace{0001}_{1}\underbrace{1111}_{F}.$$

El número hexadecimal equivalente es $0x1F = 16^1 + 15 \cdot 16^0 = 31$.

De manera similar, podemos convertir números binarios a la base **octal** (base $8=2^3$), y viceversa, esta vez tomando un conjunto de **3 dígitos binarios**, tomados de derecha a izquierda. Veamos el siguiente ejemplo donde tomamos exactamente el mismo número binario que en el Ejemplo 1.5.

Ejemplo 1.6

Un número binario $1\ 1111 = 2^4 + 2^3 + 2^2 + 2^1 + 1 = 31$ *convertido a octal:*

$$\underbrace{011}_{3}\underbrace{111}_{7}.$$

El número octal equivalente es $(37)_8 = 3 \cdot 8^1 + 7 \cdot 8^0 = 31$.

1.3. Operaciones lógicas básicas

1.3.1. Valores lógicos o valores de verdad

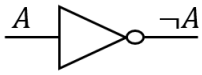
En una computadora, todo es cálculo sobre números binarios. Como sólo podemos manipular 1's y 0's, se dice que manipulamos valores lógicos. De hecho, la computadora sólo realiza operaciones lógicas (¡incluso cuando hace operaciones aritméticas!). De esta forma, las **operaciones lógicas producen valores lógicos**, también llamados **valores de verdad**:

valor lógico	valor de verdad
1	`verdadero` (V)
0	`falso` (F)

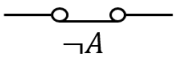
Existen 3 operaciones lógicas básicas: NO (NEGACIÓN), Y (CONJUNCIÓN) y O (DISYUNCIÓN). Cada una de ellas opera como una función que recibe uno o más valores de verdad a la entrada y produce un sólo valor de verdad a la salida. A continuación describimos estas operaciones.

1.3.2. Negación (NO)

La NEGACIÓN (NOT en inglés), es una función que toma un sólo valor de verdad como entrada y produce su negación (valor contrario) a la salida. Usaremos la siguiente escritura para representar la negación de A : $\neg A$. La Figura 1.2 presenta su simbología esquemática y la analogía de su operación mediante un interruptor normalmente cerrado.



a) NEGACIÓN (NOT)



b) Un interruptor normalmente cerrado, equivalente a la operación NOT

Figura 1.2: a) Compuerta NEGACIÓN (NOT). b) Interruptor normalmente cerrado como representación de la operación de la compuerta. El estado normalmente cerrado indica que la operación de la compuerta es tal que la salida es 1 cuando A=0 (cerrado), y es 0 cuando A=1 (abierto).

Para expresar todos los valores posibles que puede arrojar la función, usamos **una Tabla de Verdad**.

Definición 1.4. Tabla de Verdad. Una tabla de verdad es una tabla donde colocamos todas las combinaciones posibles de valores lógicos que recibe una función lógica (entrada) y el valor lógico que produce la función como resultado de la operación lógica sobre cada entrada (salida).

La tabla de verdad de la NEGACIÓN es la siguiente:

Tabla 1.1: Tabla de verdad de la NEGACIÓN.

entrada	salida
A	$\neg A$
V	F
F	V

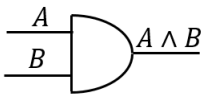
Ejemplo 1.7

Por ejemplo, podemos aplicar la negación a un número binario:

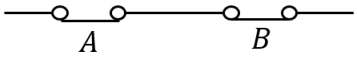
NO (11001011) = 00110100

1.3.3. Conjunción (Y)

La operación lógica de la conjunción (Y) recibe dos valores lógicos a la entrada y produce un sólo valor a la salida. Usaremos la siguiente escritura para representar la conjunción (A Y B) : $A \wedge B$. Su operación está fundamentada en que, si ambos valores de entrada son verdaderos, entonces el valor que resulta de la conjunción también es verdadero. La Figura 1.3 muestra su diagrama esquemático y analogía gráfica usando interruptores.



a) CONJUNCIÓN (AND)



b) Dos interruptores normalmente abiertos en serie, equivalentes a la operación AND

Figura 1.3: a) Compuerta CONJUNCIÓN (Y). b) Dos Interruptores normalmente abiertos, conectados en serie como representación de la operación de la compuerta. La seriación indica que la operación de la compuerta es tal que la salida es 1, sólo cuando A=1 (cerrado) y B=1 (cerrado).

La tabla de verdad de la CONJUNCIÓN es la siguiente:

Tabla 1.2: Tabla de verdad de la CONJUNCIÓN.

entrada		salida
A	B	$A \wedge B$
V	V	V
V	F	F
F	V	F
F	F	F

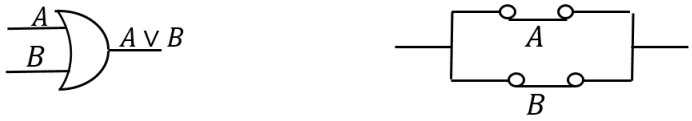
Ejemplo 1.8

Por ejemplo, podemos calcular la conjunción de dos números binarios:

$$\begin{array}{cc} & 11110010 \\ Y & 11001011 \\ \hline & 11000010 \end{array}$$

1.3.4. Disyunción (O)

La operación lógica de la disyunción (O), está fundamentada en que, dados dos valores lógicos a la entrada, si al menos uno de ellos es verdadero, entonces su disyunción es verdadera. Usaremos la siguiente escritura para representar la disyunción (A O B) : $A \vee B$. La Figura 1.4 muestra su diagrama esquemático y analogía gráfica usando interruptores.



c) DISYUNCIÓN (OR)

d) Dos interruptores normalmente abiertos en paralelo, equivalentes a la operación OR

Figura 1.4: a) Compuerta DISYUNCIÓN (O). b) Dos Interruptores normalmente abiertos, conectados en paralelo como representación de la operación de la compuerta. El paralelismo del estado normalmente abierto indica que la operación de la compuerta es tal que la salida es 1, en aquellos casos cuando A=1 (cerrado) o B=1 (cerrado).

La tabla de verdad de la DISYUNCIÓN es la siguiente:

Tabla 1.3: Tabla de verdad de la DISYUNCIÓN.

entrada	salida
A B	$A \vee B$
V V	V
V F	V
F V	V
F F	F

Ejemplo 1.9

$$\begin{array}{rr} & 11110010 \\ O & 11001011 \\ \hline & 11111011 \end{array}$$

Ejercicio Resuelto 1.3. Determine el resultado de las siguientes operaciones lógicas sobre números binarios.

1. NO (0111 1011)

2. 1110 Y 0111
3. 110001 O 001110
4. 001100 Y 01110

SOLUCIÓN

1. NO (0111 1011) = 1000 0100
2. 1110 Y 0111 = 0110
3. 110001 O 001110 = 111111
4. 001100 Y 01110 = 001100 Y 0|01110 = 001100

1.4. Ejercicios y Problemas

1. Diga si el número es correcto dada la base que se proporciona. Justifique brevemente su respuesta.
   a) base 2: -1
   b) base 6: 123454321
   c) base 4: 332711
   d) base 9: $7788A$
   e) base 11: $23A998$
   f) base 16: $16FF\ FFFF\ FFFF\ FFFF$
   g) base 17: $ABCG12$
2. Calcule el número decimal equivalente.
   a) $(1\,0000\,0000)_2$
   b) $(11011\,10010)_2$
   c) $(0010\,0111)_5$
   d) $(02\,1070)_8$
   e) $(A00)_{16}$
3. Considere las técnicas de conversión vistas en la Sección 1.2.4.
   a) Explique por qué estas técnicas funcionan.

b) Utilice estas técnicas para realizar las siguientes conversiones:

1) $(1\,1010\,0101\,1101)_2 \longrightarrow N_{16}$
2) $(11011\,10010)_2 \longrightarrow N_8$
3) $(3724)_8 \longrightarrow N_2$
4) $(1032)_4 \longrightarrow N_2$

4. Considere números binarios de n bits.

a) ¿Cuál sería el número más grande que podría representar si:

1) $n = 8$?
2) $n = 16$?

b) Suponiendo que los números negativos se representan utilizando el MSB para determinar el signo: 0 para números positivos, 1 para números negativos. Por ejemplo, en 8 bits, el número positivo más grande sería $01111111 = 127$, y el número negativo más pequeño sería $10000000 = -128$. En este caso el -1 se representa así: 11111111. Determine el número positivo más grande y el número negativo más pequeño que se puede representar con números de 16 bits.

5. El tamaño de una memoria RAM en una computadora se mide en unidades de bytes. Por ejemplo, una memoria que puede almacenar 1024 bytes, se dice que es una memoria de 1KB (1 Kilo-Byte), una memoria de $1024 \times 1024 = 1,048,576$ bytes, se dice que almacena 1MB (1 Mega-Byte). Suponga que cuenta con una memoria de 1KB. Sabiendo que un caracter cualquiera ocupa en memoria 1 byte:

a) ¿Cuantos caracteres caben en su memoria?

b) ¿De qué tamaño debe elegir su memoria (en unidades enteras de KB o MB) si desea almacenar 10000 imágenes de 24x24 pixeles, suponiendo que cada pixel se puede representar usando caracteres?

6. Realice las siguientes operaciones lógicas. Declare todas las suposiciones que haga.

a) NO (1111)

b) (010 1010) O (101 0101)

c) (NO(1110)) Y (1111)

d) (11 0111 0010) Y (110 0110)

e) (1010 0000) O (NO(0011 0011))

f) (1110 Y 0110) O 0011

g) (1110 Y 0110) O NO (0010)

7. Para cada uno de las siguientes expresiones lógicas, dibuje el circuito lógico equivalente. Suponga que las letras mayúsculas ($A, B, \ldots$) se refieren a cualquier variable binaria.

   a) (NO(A)) O (NO(B))

   b) (B Y (NO(A) O C))

   c) NO(A Y C) Y NO(D O B)

   d) NO(NO(A) Y B) Y C

   e) NO ((A Y B) Y C) Y D

8. Existe una operación lógica llamada XOR (O exclusivo), que sólo arroja verdadero cuando los números binarios a la entrada son de valor lógico distinto uno de otro; es decir, la salida es 0 cuando ambos valores de entrada son iguales.

   a) Construya la Tabla de Verdad del XOR.

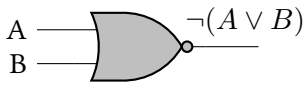
   b) Diibuje el circuito lógico de XOR usando sólo compuertas NOT, AND y OR.

   c) ¿Qué interés práctico puede tener esta operación lógica? Proporcione un ejemplo.

9. [12 pts.] Existe una compuerta lógica NOR (NO-O) que es muy útil para el diseño de circuitos lógicos. Su tabla de verdad es la siguiente:

entrada		NOR
A B	$A \vee B$	$\neg(A \vee B)$
V V	V	F
V F	V	F
F V	V	F
F F	F	V

   Su circuito lógico es el siguiente:

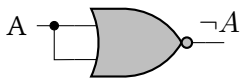


   Para cada una de las siguientes operaciones lógicas escriba la tabla de verdad de la operación mostrada, utilizando sólo la operación NOR, y dibuje el circuito lógico equivalente usando sólo compuertas NOR. Se muestra el ejemplo para la operación lógica NO(A).

   NO(A): Tabla de Verdad usando NOR:

entrada	NOR		
A	$A \vee A$	$\neg(A \vee A)$	$\neg A$
V	V	F	F
F	F	V	V

NO(A): Circuito lógico usando NOR.



Para resolver los siguientes problemas, puedes apoyarte en las siguientes equivalencias, llamadas leyes de De Morgan:

1) $\neg(A \wedge B) \equiv \neg A \vee \neg B$
2) $\neg(A \vee B) \equiv \neg A \wedge \neg B$

a) A Y B

b) A O NO(B)

10. La operación de suma de dos números binarios de 1 bit arroja los siguientes resultados $0 + 0 = 0$, $0 + 1 = 1$, $1 + 0 = 1$, $1 + 1 = 10$. La última operación se podría escribir $1 + 1 = 1|\,0$, apartando el MSB en el resultado (usando $|$) porque en realidad se trata de un bit adicional, llamado "acarreo"(*carry bit* en inglés).

    a) Construya la Tabla de Verdad de este sumador usando sólo operaciones lógicas (NO, Y, O). Considere el bit de acarreo como una salida en 0 inicialmente.

    b) Escriba la expresión lógica del resultado de la suma.

    c) Igual para el resultado del acarreo.

    d) Modifique la tabla de verdad que acaba de construir para un sumador de números binarios de 2 bits. NOTA: necesita considerar un bit de acarreo intermedio, ya que, por ejemplo: $01 + 01 = 0|10$, y $11 + 01 = 1|00$.

    e) Escriba ahora las expresiones lógicas de un sumador de 2 bits (resultado y acarreos).

Capítulo 2

Lógica Proposicional

Jorge Hermosillo Valadez

2.1. ¿Qué es la Lógica Proposicional?

La **Lógica Proposicional** es una disciplina de la ciencia computacional que permite estudiar sistemáticamente la **deducción lógica**. En este sentido, se trata de una **herramienta matemática de expresión y análisis de la argumentación y el razonamiento lógicos**.

Es parte de lo que se conoce como el **Pensamiento Computacional**, su importancia radica en el hecho de que el diseño de algoritmos, así como su programación mediante algún lenguaje de computadora y su validación a la hora de la ejecución, pasan por la criba del razonamiento lógico.

2.2. Noción de proposición lógica

Definición 2.1. **Proposición lógica**. En el contexto de la ciencia de la computación, una proposición es *una afirmación o frase declarativa a la que se le puede atribuir un valor de verdad de verdadero o falso, pero no ambos.*

En este contexto, la **semántica** de una proposición lógica se refiere a **su significado en términos de valores de verdad**. Es decir, el significado de una proposición sólo puede ser `verdadero` o `falso`, pero no ambos a la vez.

Ejemplo 2.1

Por ejemplo, las siguientes son proposiciones válidas:

1. *"Juan es un estudiante."*
2. *"Hoy lloverá."*
3. *"Los marcianos son de color verde."*
4. *"El transporte llegó tarde."*

Las siguientes frases *no* son proposiciones válidas:

frase	justificación
"No zapatos, no camisa, no servicio"	No hay verbo y las frases no están articuladas.
"¡Qué buena película!"	Las expresiones de exclamación no son válidas.
"¿Qué hora es?"	Las preguntas no afirman.

Para efectos de la lógica proposicional, toda proposición se representa usando letras minúsculas del tipo:

$$p, q, r, s$$

Ejemplo 2.2

Por ejemplo, las proposiciones del Ejemplo 2.1 se pueden simbolizar de la siguiente forma:

A este tipo de proposiciones se les conoce como **proposiciones atómicas** o simplemente **átomos**.

- p : *"Juan es un estudiante."*
- q : *"Hoy lloverá."*
- r : *"Los marcianos son de color verde."*
- s : *"El transporte llegó tarde."*

.

Observación 2.1. Note que, de manera similar a las operaciones lógicas vistas anteriormente, podemos hacer **cálculo sobre proposiciones lógicas**, puesto que podemos atribuirles valores de verdad. En este sentido, el cálculo proposicional se refiere al estudio de los valores de verdad que guarda la composición de proposiciones atómicas usando conectivos lógicos.

2.3. Conectivos básicos

Los conectivos básicos se refiere a los símbolos que ya hemos utilizado para realizar operaciones lógicas entre números, a los que agregaremos uno más como se muestra en la siguiente tabla.

Tabla 2.1: Conectivos básicos de la Lógica Proposicional.

Conectivo	Nombre
$\neg$	`negación`
$\wedge$	`conjunción`
$\vee$	`disyunción`

A continuación definimos estos conectivos en el contexto de la Lógica proposicional. Las Tablas de verdad correspondientes son las Tablas 1.1 a 1.3.

Definición 2.2. Sea p una proposición. La *negación de* p, denotada por $\neg p$, es la declaración

"No es el caso que p"

La proposición $\neg p$ se lee "no p". El valor de verdad de la negación de p, $\neg p$, es el opuesto al valor de verdad de p.

La negación de una proposición también puede considerarse el resultado de la operación del operador de negación sobre una proposición.

Ejemplo 2.3

A continuación se listan algunos ejemplos de proposiciones y su negación respectiva (literal) y una o más expresiones coloquiales equivalentes:

1. *"Juan está comiendo una manzana."*
   - ▷ *"No es el caso que Juan está comiendo una manzana."*
   - ▷ *"Juan no está comiendo una manzana."*
2. *"Esta computadora tiene al menos 16GB de RAM."*
   - ▷ *"No es el caso que esta computadora tiene al menos 16GB de RAM."*
   - ▷ *"Esta computadora no tiene al menos 16 GB de RAM."*
   - ▷ *"Esta computadora tiene menos de 16 GB de RAM."*

Definición 2.3. Sean p y q dos proposiciones. La *conjunción de* p y q, denotada por $p \wedge q$, es la proposición "p y q". La conjunción $p \wedge q$ es verdadera, cuando ambas p y q son verdaderas, y es falsa en caso contrario.

Ejemplo 2.4

Por ejemplo, se muestran a continuación dos proposiciones y su conjunción:

1. *"Andrea tiene una computadora con un disco duro de más 250GB."*
2. *"El procesador de la computadora de Andrea corre a más de 1GHz."*
   - ▷ *"Andrea tiene una computadora con un disco duro de más 250GB, y el procesador de la computadora de Andrea corre a más de 1GHz."*
   - ▷ *"Andrea tiene una computadora con un disco duro de más 250GB, y su procesador corre a más de 1GHz."*

Definición 2.4. Sean p y q dos proposiciones. La *disyunción de* p y q, denotada por $p \vee q$, es la proposición "p o q". La disyunción $p \vee q$ es falsa, cuando ambas p y q son falsas, y es verdadera en caso contrario.

El uso del conectivo "o" en una disyunción corresponde a una de las dos formas en que se usa la palabra o en español, a saber, como un **"o inclusivo"**. **Una disyunción es verdadera cuando al menos una de las dos proposiciones es verdadera**. Por ejemplo, el o inclusivo se usa en la declaración

"Los estudiantes que han tomado cálculo o ciencias de la computación pueden tomar esta clase".

Aquí, queremos decir que los estudiantes que han tomado tanto cálculo como computación pueden tomar la clase, así como los estudiantes que han tomado solo alguna de las dos materias.

Por otro lado, existe el o exclusivo cuando decimos

"Los estudiantes que hayan tomado cálculo o ciencias de la computación, pero no ambos, pueden inscribirse en esta clase".

Aquí, queremos decir que los estudiantes que han tomado ambos cursos, cálculo y computación, no pueden tomar la clase. Solo aquellos que hayan tomado exactamente uno de los dos cursos pueden tomar la clase.

De manera similar, cuando un menú en un restaurante dice: "El desayuno viene con jugo o fruta", el restaurante casi siempre quiere decir que los clientes pueden pedir jugo o fruta, pero no ambos.

Ejemplo 2.5

Por ejemplo, se muestra a continuación la disyunción de las proposiciones del Ejemplo 2.4:

- ▷ *"Andrea tiene una computadora con un disco duro de más 250GB, o el procesador de la computadora de Andrea corre a más de 1GHz."*

Esta proposición es verdadera cuando la computadora de Andrea tiene un disco duro de más 250GB, cuando el procesador de su computadora corre a más de 1GHz y cuando ambas de estas condiciones son verdaderas. Es falsa cuando ambas condiciones son falsas, es decir, cuando la computadora de Andrea tiene un disco duro de menos de 250GB ***y*** *el procesador de su computadora funciona a 1 GHz o menos.*

2.4. Proposiciones condicionales

Veremos ahora uno de los conectivos de mayor utilidad en Lógica, y en general en la ciencia de la computación.

Definición 2.5. Sean p y q dos proposiciones. La *declaración condicional* $p \to q$ es la proposición "si p entonces q". La declaración condicional $p \to q$ es falsa, cuando p es verdadera y q es falsa, y es verdadera en caso contrario. En la declaración condicional $p \to q$, a p se le llama **hipótesis** (o **antecedente**, o **premisa** o **condición suficiente**), y a q se le conoce como **conclusión** (o **consecuencia** o **condición necesaria**).

A continuación veremos la Tabla de verdad de la implicación.

Tabla 2.2: Tabla de verdad de la IMPLICACIÓN.

p	q	$p \to q$
V	V	V
V	F	F
F	V	V
F	F	V

Desde un **punto de vista lógico**, el conectivo **implicación** es la base de un razonamiento condicional que se podría enunciar de la siguiente forma:

$$p \to q$$

A condición de que la hipótesis p sea verdadera, la conclusión q es verdadera.

Que se podría enunciar también así:

Observación 2.2.

$$p \to q \equiv \neg q \to \neg p$$

Si la conclusión q es falsa, la hipótesis p también es falsa.

Debido a que las declaraciones condicionales juegan un papel esencial en el razonamiento matemático, se usa una variedad de terminología para expresar $p \to q$. Las siguientes son formas de expresar esta declaración condicional:

Tabla 2.3: Formas comunes de expresar la implicación $p \to q$.

"si p, entonces q"	"p implica q"
"si p, q"	"p solo si q"
"p es suficiente para q"	"una condición suficiente para q es p"
"q si p"	"q siempre que p"
"q cuando p"	"q es necesario para p"
"una condición necesaria para p es q"	"q se sigue de p"
"q a menos que ¬p"	

2.5. Proposiciones compuestas

Los conectivos lógicos permiten crear proposiciones lógicas compuestas a partir de proposiciones atómicas. Por ejemplo, consideremos las siguientes proposiciones en lenguaje natural:

1. "Si hoy llueve y no llevo paraguas, me voy a mojar."
2. "Quiero postre y café, poco dulce."
3. "Los archivos con tamaño menor a 1MB se podrán subir a la plataforma, siempre y cuando estén en formato pdf o en algún otro formato pero no contengan imágenes."

4. “La obtención de la ficha no significa la admisión directa."

Consideremos ahora los siguientes átomos:

1. p: “hoy llueve"; q: “llevo paraguas"; r: “me voy a mojar"
2. p: “quiero postre"; q: “quiero café"; q: “el café es poco dulce"
3. p: “Los archivos con tamaño menor a 1MB se podrán subir a la plataforma"; q: “Los archivos están en pdf"; r: “Los archivos contienen imágenes."
4. p: “Alguien obtiene la ficha"; q: “Se logra la admisión directa."

Podemos simbolizar estas proposiciones usando estos átomos de la siguiente forma:

Note que proposiciones simbólicas del estilo: (p, q) o $(\neg p \neg q)$ o $(p \wedge \vee q)$ o $(p \neg \rightarrow q)$, son incorrectas, y por ende no son admisibles.

1. $p \wedge \neg q \rightarrow r$
2. $p \wedge q \wedge r$
3. $(q \vee (\neg q \wedge \neg r)) \rightarrow p$
4. $\neg(p \rightarrow q)$

.

2.6. Ejercicios y Problemas

1. Diga cuáles de las siguientes frases son proposiciones y cuáles son sus valores de verdad.
   a) La UAEM está en el estado de Morelos.
   b) Cuernavaca es conocida como “la ciudad de la eterna primavera”.
   c) Responda a esta pregunta.
   d) Acapulco es la capital de México.
   e) Los perros pueden volar.
   f) 3 x 3 = 9.
   g) 2 + 3 = 4.
   h) 4 + x = 5.
2. Diga cual es la negación de las siguientes proposiciones.

a) Ciro tiene una pera loca.

b) No hay contaminación en la Cd. de México.

c) Se ha dicho que el béisbol es el rey de los deportes.

d) La primavera en Cuernavaca es calurosa y soleada.

e) Tamara envía más de 100 mensajes al día.

f) $8 \cdot 7 \cdot 11 = 560$

3. Supongamos que el teléfono celular A tiene 256MB de RAM, 32GB de ROM, y la resolución de su cámara es de 8 MP; el teléfono B tiene 288 MB de RAM, 64 GB de ROM, y el la resolución de su cámara es de 4 MP; y el celular C tiene 128 MB de RAM, 32 GB de ROM, y la resolución de su cámara es de 5 MP. Determine el valor de verdad de cada una de las siguientes proposiciones:

   a) El teléfono B tiene la mayor cantidad de RAM de estos tres celulares.

   b) El celular C tiene más ROM o una cámara con resolución más alta que el teléfono B.

   c) El teléfono B tiene más RAM, más ROM y una cámara de mayor resolución que el celular A.

   d) Si el celular B tiene más RAM y más ROM que el teléfono C, entonces también tiene una cámara con resolución más alta.

   e) Si la resolución de cámara del celular A es más baja que la resolución de C, entonces el celular A no tiene más RAM que C.

4. Sean p y q las siguientes proposiciones:

   p : Compré un billete de lotería esta semana.

   q : Gané el premio mayor.

   Exprese cada una de estas proposiciones compuestas como una frase en español.

   a) $\neg p$ *b*) $p \vee q$ *c*) $p \rightarrow q$
   d) $p \wedge q$ *e*) $\neg p \rightarrow \neg q$ *f*) $\neg p \wedge \neg q$
   g) $\neg p \vee (p \wedge q)$

5. Sean p y q las siguientes proposiciones:

   p : Manejas a más de 110 Km/h.

   q : Tienes una multa por exceso de velocidad.

   Simbolice las siguientes proposiciones compuestas usando conectivos.

a) No manejas a más de 110 Km/h.

b) Manejas a más de 110 Km/h, pero no tienes una multa por exceso de velocidad.

c) Tendrás una multa por exceso de velocidad si manejas a más de 110 Km/h.

d) Si no manejas a más de 110 Km/h, entonces no tendrás una multa por exceso de velocidad.

e) Manejar a más de 110 Km/h es suficiente para tener una multa por exceso de velocidad.

f) Tienes una multa por exceso de velocidad, pero no manejas a más de 110 Km/h.

g) Cuando tienes una multa por exceso de velocidad, estás manejando a más de 110 Km/h.

6. Sean p y q las siguientes proposiciones:

   p : La humedad es mayor a 80 %.

   q : Hace mucho calor.

   Simbolice las siguientes proposiciones compuestas usando conectivos.

   a) La humedad es mayor a 80 % y hace mucho calor.

   b) La humedad es mayor a 80 % pero no hace mucho calor.

   c) La humedad no es mayor a 80 % y no hace mucho calor.

   d) Hace mucho calor o la humedad es mayor a 80 % (o ambas).

   e) Si la humedad es mayor a 80 %, también hace mucho calor.

   f) La humedad es mayor a 80 % o hace mucho calor, pero no hace mucho calor si la humedad es mayor a 80 % .

7. Sean p, q y r las siguientes proposiciones:

   p : Tienes gripe.

   q : Faltas al último examen.

   r : Pasas el curso.

   Exprese cada una de estas proposiciones compuestas como una frase en español.

   a) $p \to q$ *b)* $(\neg q \to r) \wedge (r \to \neg q)$ *c)* $q \to \neg r$
   d) $p \vee q \vee r$ *e)* $(p \to \neg r) \vee (q \to \neg r)$ *f)* $(p \wedge q) \vee (\neg q \wedge r)$

8. Determine si las siguientes condicionales son verdaderas o falsas.

   a) Si 1 + 1 = 2, entonces 2 + 2 = 5.

b) Si 1 + 1 = 3, entonces 2 + 2 = 4.

c) Si 1 + 1 = 3, entonces 2 + 2 = 5.

d) Si 1 + 1 = 2, entonces los perros pueden volar.

e) Si 1 + 1 = 3, entonces los perros pueden volar.

f) 2 + 2 = 4 sí y sólo sí 1 + 1 = 2.

g) 0 >1 sí y sólo sí 2 >0.

9. Sean p, q y r las siguientes proposiciones:

   p : Se han visto pumas en la zona.

   q : Caminar por el sendero es seguro.

   r : Las moras están maduras a lo largo del camino.

   Simbolice las siguientes proposiciones compuestas usando conectivos. Use paréntesis donde considere que es necesario para dejar en claro la delimitación de cada proposición.

   a) Las moras están maduras a lo largo del camino pero no se han visto pumas en la zona.

   b) No se han visto pumas en la zona y caminar por el sendero es seguro, pero las moras están maduras a lo largo del camino.

   c) Si las moras están maduras a lo largo del camino, caminar es seguro siempre y cuando no se hayan visto pumas en la zona.

   d) No es seguro caminar por el sendero, pero no se han visto pumas en la zona y las moras están maduras a lo largo del camino.

   e) Para que caminar por el sendero sea seguro, es necesario pero no suficiente que las moras no estén maduras a lo largo del camino y que no se hayan visto pumas en la zona.

   f) Caminar por el sendero no es seguro cuando se han visto pumas en la zona y las moras están maduras a lo largo del camino.

10. Para cada una de las siguientes frases, determine si se trata de un o inclusivo, o, un o exclusivo. Justifique su respuesta.

    a) Se requiere experiencia en C++ o Java.

    b) El paquete incluye sopa o ensalada.

    c) Para entrar al país se requiere pasaporte o credencial de elector.

    d) Publicar o perecer.

11. Escriba cada una de las siguientes declaraciones en la forma “si p, entonces q” en español. Observe la tabla de formas comunes de enunciar declaraciones condicionales.

    a) Es necesario trabajar duro para titularse.

    b) Vientos del sur implican brisa del mar.

    c) Una condición suficiente para hacer válida la garantía es que haya comprado la computadora hace menos de un año.

    d) A Rodrigo lo pescan cada vez que hace trampa.

    e) Puedes acceder al sitio web sólo si pagas tu cuota de suscripción.

    f) Ser elegido es consecuencia de conocer las personas adecuadas.

    g) Carolina se marea siempre que sube a un bote.

Capítulo 3

Pensamiento Lógico

Jorge Hermosillo Valadez

3.1. Análisis de proposiciones compuestas

¿Cómo saber el valor de verdad de una proposición compuesta?, es decir, ¿cómo evaluar una proposición lógica en general? En ciencia computacional, esta pregunta tiene que ver con la estrecha **relación que existe entre sintaxis y semántica**: entre la estructura y el ordenamiento de los símbolos de una proposición lógica y su valor de verdad.

En el ámbito de la programación, esta pregunta está relacionada con el enrutamiento del procesamiento, o cómputo, luego de la valoración de expresiones lógicas: *qué acción ejecutar, en función del valor de verdad de una expresión lógica.*

En esta sección, usaremos convenciones sintácticas (de escritura) y tablas de verdad. Comenzaremos con un ejemplo. A partir de ahora, asumiremos que la simbolización ha sido resuelta, es decir, nos ocuparemos sólo de los símbolos, sin necesidad de saber lo que están representando en lenguaje natural.

3.2. Convenciones de alcance vinculante

Consideremos la siguiente proposición compuesta:

$$p \wedge \neg q \rightarrow r \vee \neg s \tag{3.1}$$

La primera pregunta es ¿cómo leer 3.1? Esta pregunta es relevante en el contexto del diseño de condiciones de bifurcación en algoritmos.

Para responder a esta pregunta, utilizaremos las siguientes convenciones de **alcance vinculante** de los conectivos, mostrados en la Tabla 3.1, las cuales nos permiten saber, en ausencia de paréntesis, hasta dónde aplica un conectivo; es decir: qué átomos o proposiciones compuestas está relacionando un conectivo en ausencia de paréntesis.

Tabla 3.1: Convención de alcance vinculante de conectivos.

conectivo	alcance vinculante
$\neg$	Sólo se vincula con el símbolo inmediato siguiente
$\wedge$	Vincula a los símbolos inmediatos a la izquierda y derecha
$\vee$	Vincula a los símbolos inmediatos a la izquierda y derecha
$\rightarrow$	Vincula a todos los símbolos a la izquierda y a la derecha

3.3. Semántica de proposiciones compuestas

La Tabla 3.1 nos dice que el conectivo de negación tiene el menor alcance vinculante, lo que significa que su efecto aplica al símbolo inmediato siguiente. Por ejemplo, en la siguiente expresión la negación sólo aplica al átomo p:

$$\neg p \vee q \rightarrow r$$

Si quisiéramos negar la disyunción completa, tendríamos que usar paréntesis:

$$\neg(p \vee q) \rightarrow r$$

Si ahora, quisiéramos negar toda la implicación, tendríamos que escribir:

$$\neg(p \vee q \rightarrow r)$$

Enseguida, la Tabla 3.1 indica que los conectivos de conjunción y disyunción tienen el mismo alcance vinculante. Los siguientes ejemplos muestran que la conjunción y disyunción aplican a cada par de símbolos por igual, no habiendo conflicto desde el punto de vista semántico:

$$p \wedge q \wedge r$$

$$p \vee q \vee r$$

Si quisiéramos combinar conjunciones con disyunciones, estamos obligados a usar paréntesis, ya que de otra forma sería imposible saber el orden de asociación de los átomos:

$$(p \wedge q) \vee r$$
$$p \wedge (q \vee r)$$

Una tabla de verdad nos permitiría darnos cuenta de que las expresiones anteriores son semánticamente distintas. Se deja la verificación de esta afirmación para el Ejercicio 3.7.2.

Finalmente, el conectivo de implicación tiene el mayor alcance vinculante, como lo muestran los siguientes ejemplos, donde la implicación se da entre la conjunción del lado izquierdo y la disyunción del lado derecho:

$$p \wedge q \wedge r \rightarrow s \vee t \vee u$$

Si se deseara, por ejemplo, relacionar sólo los átomos r y s con la implicación, se podría escribir:

$$p \wedge q \wedge (r \rightarrow s) \vee t \vee u$$

Sin embargo, esta proposición es **semánticamente ambigua**, ya que la proposición resultante es una combinación de conjunciones con disyunciones, por lo que se debe tomar una decisión acerca de dónde colocar los paréntesis. Aquí, hay muchas opciones. Por ejemplo, si se desea relacionar conjuntivamente la implicación con p y q, entonces se debe escribir:

$$(p \wedge q \wedge (r \rightarrow s)) \vee t \vee u \tag{3.2}$$

Otra alternativa es si se desea relacionar disyuntivamente la implicación con t o u, en cuyo caso se debe escribir:

$$p \wedge q \wedge ((r \rightarrow s) \vee t \vee u) \tag{3.3}$$

Se deja la evaluación de expresiones similares usando tablas de verdad para el Ejercicio 3.7.3.

El asunto de cómo colocar paréntesis es de suma importancia, como se habrá podido observar de los ejemplos anteriores, ya que eso determina la semántica, el significado, en términos de valores de verdad, de una expresión o proposición lógica. Esto es tanto más importante cuanto el resultado de un proceso de cómputo depende, no sólo de una evaluación correcta de la expresión, sino de que la expresión misma refleje exactamente la condición que se desea identificar o capturar.

Ejercicio Resuelto 3.1. Para cada uno de los siguientes incisos, haga la tabla de verdad de la expresión lógica.

1. $\neg r \vee q$
2. $\neg(q \vee p)$
3. $\neg(p \wedge q) \vee r$
4. $\neg p \rightarrow r \vee q$

SOLUCIÓN

1. $\neg r \vee q$

r	q	$\neg r$	$\neg r \vee q$
V	V	F	V
V	F	F	F
F	V	V	V
F	F	V	V

2. $\neg(q \vee p)$

p	q	$q \vee p$	$\neg(q \vee p)$
V	V	V	F
V	F	V	F
F	V	V	F
F	F	F	V

3. $\neg(p \wedge q) \vee r$

p	q	r	$p \wedge q$	$\neg(p \wedge q)$	$\neg(p \wedge q) \vee r$
V	V	V	V	F	V
V	V	F	V	F	F
V	F	V	F	V	V
V	F	F	F	V	V
F	V	V	F	V	V
F	V	F	F	V	V
F	F	V	F	V	V
F	F	F	F	V	V

4. $\neg p \rightarrow r \vee q$

p	q	r	$\neg p$	$r \vee q$	$\neg p \to r \vee q$
V	V	V	F	V	V
V	V	F	F	V	V
V	F	V	F	V	V
V	F	F	F	F	V
F	V	V	V	V	V
F	V	F	V	V	V
F	F	V	V	V	V
F	F	F	V	F	F

3.4. Noción de equivalencia semántica

De manera intuitiva, la equivalencia semántica entre dos proposiciones se da cuando sus valores de verdad son los mismos en una tabla de verdad.

Ejemplo 3.1

Por ejemplo, las proposiciones $(p \to q)$ *y* $(\neg p \vee q)$ *son semánticamente equivalentes.*

p	q	$p \to q$	$\neg p \vee q$
V	V	V	V
V	F	F	F
F	V	V	V
F	F	V	V

La noción de equivalencia semántica es muy importante en matemáticas y computación, ya que permite identificar mecanismos de transformación de una proposición a otra equivalente, con el fin de encontrar una **tautología** o una **contradicción**.

Definición 3.1 (Tautología)**.** Una tautología es una proposición que es verdadera siempre, independientemente del valor de verdad de sus átomos. Por ejemplo:

$$p \vee \neg p.$$

Definición 3.2 (Contradicción)**.** Una contradicción es una proposición que es falsa siempre, independientemente del valor de verdad de sus átomos. Por ejemplo:

$$p \wedge \neg p.$$

La contradicción es la negación de la tautología.

3.5. Razonamiento deductivo

La base de pensamiento lógico es el razonamiento, o argumentación. Un argumento es un razonamiento que se emplea para probar o demostrar una proposición, o bien para convencer a alguien de aquello que se afirma o se niega. En nuestro contexto, debemos pues entender que un argumento es una secuencia de proposiciones, que se encadenan en un razonamiento lógico que termina en una conclusión, siendo ésta el objetivo del argumento.

La construcción de argumentos correctos, o válidos, es de gran importancia en el diseño de algoritmos. En el siguiente capítulo se abordará el tema de los algoritmos con más detenimiento. Por ahora, diremos que un algoritmo es una secuencia ordenada de pasos para resolver un problema. De esta forma, la resolución de problemas mediante algoritmos involucra una serie de acciones consecutivas que persiguen un objetivo. En este sentido, se puede ver a un algoritmo como una secuencia de proposiciones, que forman parte de un razonamiento lógico: cada paso en el algoritmo debe ser una consecuencia lógica de un paso anterior.

Por otra parte, y no menos importante, es muy común en la práctica, que un algoritmo falle por un error de lógica. Por lo tanto, es de suma importancia en el diseño de soluciones algorítmicas a un problema entender qué es un argumento válido, y cómo construir condiciones lógicas correctamente.

3.5.1. Deducción e inducción

Todo argumento lógico parte de un cierto número de **premisas** y llega a una **conclusión**.

Definición 3.3. Premisa. Una premisa es una proposición que desde un principio se asume como verdadera. En matemáticas se conoce como axioma. Es el punto de partida de todo razonamiento lógico. Muchas veces, partimos de premisas que *suponemos* verdaderas. A estas se les llama hipótesis.

Veamos un ejemplo de argumento.

Ejemplo 3.2

El siguiente argumento tiene dos premisas (1 y 2) y una conclusión (3).

1. *Todo hombre es mortal.*
2. *Sócrates es un hombre.*
3. *Sócrates es mortal*

Este argumento es lo que se conoce como un **silogismo**.

Definición 3.4 (Silogismo)**.** Un silogismo es un razonamiento deductivo que tiene 2 premisas y una conclusión.

La experiencia humana, nos diría que el argumento 3.2 es correcto sin mucha dificultad; diríamos incluso que es propio del sentido común. No obstante, algunos argumentos son más "fuertes" que otros. En el contexto de la ciencia computacional, hablamos de **inferencia**, para referirnos al proceso de obtener nuevas proposiciones a partir de premisas o hipótesis. Este proceso, suele categorizarse de varias maneras, dos de las más comunes son la **deducción** y la **inducción**. En otras palabras, la inferencia puede ser deductiva o inductiva.

Por **deducción**, se entiende que el argumento parte de premisas verdaderas, y **sólo pueden inferirse nuevas proposiciones con base en las anteriores**. En este caso, no hay incertidumbre respecto de la veracidad de las conclusiones. El Ejemplo 3.2 es un ejemplo de **inferencia deductiva**.

Por **inducción**, debemos entender que se trata de un proceso que puede partir de ciertas evidencias aisladas para llegar a conclusiones generalizadas. Se podría decir que se trata de un proceso similar al que sigue un científico cuando realiza varias observaciones experimentales y conjetura una conclusión.

Ejemplo 3.3

Por ejemplo,

Obs. 1 Un cable de cobre conduce electricidad

Obs. 2 Un cable de plata conduce electricidad

Obs. 3 Un cable de aluminio conduce electricidad

Conc. Es altamente probable que todos los metales conduzcan electricidad.

En el contexto de la ciencia computacional, existen marcos teóricos formales para modelar en una máquina procesos de deducción e inducción. En este curso veremos sólo una introducción a lo que se conoce como deducción natural.

La primera pregunta importante es: ¿qué hace que un argumento como 3.2 sea correcto? Desde el punto de vista de la lógica proposicional, diremos que, si las premisas 1 y 2 son verdaderas, entonces la conclusión 3 es verdadera, y, por lo tanto, que el argumento es válido. Veamos esto formalmente.

3.5.2. Noción de argumento válido

La deducción lógica es mucho más propensa a la falla de lo que podríamos pensar. Veamos algunos ejemplos de silogismos falsos.

Ejemplo 3.4

El siguiente argumento es falso. Las premisas son π_1 y π_2, y la conclusión es η.

π_1 : *Fido es un perro.*

π_2 : *Todos los perros son color café.*

η : *Por tanto, Fido es color café.*

La premisa π_2 es falsa: no es verdad que todos los perros sean color café.

Aunque el argumento del Ejemplo 3.4 tiene exactamente la misma estructura que en el ejemplo de Sócrates, la falsedad de la premisa 2 falsifica el argumento en este caso.

Cualquier argumento con premisas falsas es incorrecto. En el lenguaje de la computación, este es un ejemplo de "si entra basura, sale basura".

Otra forma en que la lógica deductiva falla es cuando la conclusión no se sigue de las premisas.

Ejemplo 3.5

Por ejemplo, el siguiente argumento es falso:

π_1 : *Todas las pelotas de tenis son redondas*

π_2 : *La Tierra es redonda*

η : *Por tanto, la Tierra es una pelota de tenis*

Desde un punto de vista del sentido común, es evidente que el argumento del Ejemplo 3.5 falla. Desde el punto de vista de la lógica simbólica, este argumento dice: "Toda P es R; T es R; por tanto, T es P". Si usamos el lenguaje de la lógica proposicional, podríamos escribir:

$\pi_1 : p \to r$

$\pi_2 : t \wedge r$

$\eta : t \to p$

Esta es una forma incorrecta de deducción, por lo que el argumento es falso, en nuestro contexto, diremos que se trata de un **argumento no válido**.

¿Cómo saber sistemáticamente si un argumento es incorrecto o no? En términos generales, para saber si un argumento es válido (correcto) podemos seguir la regla siguiente:

Definición 3.5 (Argumento válido). Dadas n premisas $\pi_1, \pi_2, \ldots, \pi_n$, y una conclusión η, decimos que el argumento

$$\pi_1, \pi_2, \ldots, \pi_n \vdash \eta$$

es válido, si siempre que $\pi_1 \wedge \pi_2 \wedge \ldots \wedge \pi_n$ es verdadera, η también es verdadera.

Así, podemos mostrar que el argumento del Ejemplo 3.5, que podemos escribir:

$$\underbrace{p \to r}_{\pi_1} \ , \ \underbrace{t \wedge r}_{\pi_2} \ \vdash \ \underbrace{t \to p}_{\eta} \tag{3.4}$$

es falso, usando una tabla de verdad muy sencilla para comprobar que **no siempre** que π_1 y π_2 son ambas verdaderas, η también lo es. Se deja esta comprobación para el Ejercicio 3.7.9.

3.6. Construcción de condicionales lógicas

Existe una relación directa entre el concepto de proposición lógica y la noción de ejecución condicionada en algoritmos. En efecto, cómo se verá en el siguiente capítulo, en un algoritmo se pueden **tomar decisiones** acerca del procesamiento de los datos, **en función de una condición** que se verifica durante la ejecución del algoritmo.

Para dar cuenta de esta relación estrecha, y de la importancia de entender la construcción correcta de proposiciones lógicas daremos aquí algunas pautas.

3.6.1. Simbolización de proposiciones en lenguaje natural

Para empezar, es importante simbolizar correctamente un texto dado en lenguaje natural, en un conjunto de proposiciones atómicas, que después debemos poner en relación unas con otras. Veamos un ejemplo.

Ejemplo 3.6

> *Si la luz fuera simplemente un movimiento ondulatorio continuo, entonces la luz más brillante emitiría siempre electrones con mayor energía que los originados por luz más tenue, sin embargo, la luz más brillante no siempre emite electrones con mayor energía que los originados por luz más tenue.*

Podemos ver este ejemplo como un conjunto de oraciones que expresan un conocimiento, plantean un problema, o expresan una situación deseada. Este es un punto de partida muy común en inteligencia artificial, y en general en el análisis de problemas en ciencias computacionales. ¿Cómo proceder para simbolizar esto? La simbolización lógica de un conjunto de enunciados en lenguaje natural se le conoce también como "representación del conocimiento". Damos a continuación una pauta metodológica para pasar del lenguaje natural al lenguaje de la lógica proposicional.

Método 3.1 (Simbolización Lógica)**.** Para proceder a la representación simbólica de un texto, expresado en lenguaje natural, podemos hacer lo siguiente:

átomos: Identificar todas y cada una de las frases, u oraciones, presentes en el enunciado, con una estructura básica: *Sujeto+Verbo+Complemento*, expresada en forma de una afirmación o declaración no negada, y utilizar una letra minúscula del alfabeto, usualmente comenzando por la letra p, para representar cada una de estas frases u oraciones identificadas.

conectivos: Identificar los conectivos en el siguiente orden:

1. Negación: identificar si alguna de las proposiciones atómicas está siendo negada en el texto original.
2. Conjunción: identificar si dos, o más, proposiciones atómicas están siendo unidas por alguna conjunción como *pero, y, aunque, sin embargo, etc.*
3. Disyunción: identificar si dos, o más, proposiciones atómicas están siendo unidas por alguna disyunción como *o, o bien, etc.*
4. Implicación: identificar si dos, o más, proposiciones atómicas están siendo unidas por alguna implicación como *si... entonces, etc.*. Se recomienda consultar la Tabla 2.3.

composición: Unir todas las proposiciones atómicas, utilizando los conectivos adecuados de manera a que la semántica de la proposición compuesta se corresponda lo mejor posible con el significado de las oraciones en lenguaje natural.

Analicemos el Ejemplo 3.6 utilizando el Método 3.1.

átomos:
p : *La luz* ***es*** *simplemente un movimiento ondulatorio continuo.*
q : *La luz más brillante* ***emite*** *siempre electrones con mayor energía que los originados por luz más tenue.*

conectivos:
1. Negación: Sí hay. Se observa la negación de q.

2. Conjunción: Sí hay. Se observa una conjunción **sin embargo**.
3. Disyunción: No hay.
4. Implicación: Si hay. Se observa una implicación **si...entonces**.

composición: $(p \to q) \wedge \neg q$. Si p, entonces q, sin embargo, $\neg q$.

Veamos otro ejemplo resuelto.

Ejercicio Resuelto 3.2. Simboliza las oraciones del siguiente argumento:

Si el congreso rehúsa emitir nuevas leyes, entonces la huelga no terminará a menos que dure más de un año y el presidente de la compañía renuncie, y si el congreso emite nuevas leyes o la huelga no termina entonces durará más de un año.

átomos:
p : *El congreso **emite** nuevas leyes.*
q : *La huelga **termina**.*
r : *La huelga **dura** más de un año.*
s : *El presidente de la compañía **renuncia**.*

conectivos:
1. Negación: Sí hay. Se observa la negación de p y la negación de q.
2. Conjunción: Sí hay. Se observan dos conjunciones **y**.
3. Disyunción: Sí hay. Se observa una disyunción **o**.
4. Implicación: Sí hay. Se observan dos implicaciones **si** x **entonces** y y una más y **a menos que** $\neg x$, donde x y y son proposiciones.

composición: $\Big(\neg p \to \big(\neg(r \wedge s) \to \neg q\big)\Big) \wedge \Big((p \vee \neg q) \to r\Big)$. Si no p, entonces, si no (r y s) entonces no q, y si p o no q, entonces r.

Note que por la observación 2.2, en el ejemplo resuelto pudimos haber escrito:

$$\Big(\neg p \to \big(q \to r \wedge s\big)\Big) \wedge \Big((p \vee \neg q) \to r\Big).$$

3.6.2. Composición de proposiciones condicionales

Es muy común confundir la lógica proposicional (matemática) con la lógica causal (fenómeno físico). Por ejemplo, ¿cuál de las siguientes proposiciones es **consistente** desde un punto de vista de la Lógica Proposicional:

$$\text{Nubes} \to \text{Lluvia} \tag{3.5}$$
$$\text{Lluvia} \to \text{Nubes} \tag{3.6}$$

Si analizamos detenidamente ambas con una tabla de verdad, veremos que la proposición 3.6 es la que corresponde a una deducción lógica certera. Esto va de la mano con la Observación 2.2 que hicimos anteriormente, ya que podemos afirmar que: " si no hay nubes, entonces no hay lluvia". Es decir que, *si la proposición es verdadera, entonces la veracidad de la condición suficiente (`Lluvia`) obliga a que la condición necesaria se cumpla (`Nubes`).*

Veamos un ejemplo más cercano a una situación computacional para mostrar esto.

Ejemplo 3.7

Un profesor de Lógica desea construir una proposición que diga cuando un alumno está aprobado, o no, dadas 3 calificaciones x, y, z, que corresponden a una calificación de exámenes y de proyecto que pueden oscilar entre 5 y 10, y a un crédito por participación que puede ser verdadero o falso. Para ello, el profesor ha establecido las siguientes reglas:

1. *Si la calificación promedio de exámenes y proyecto es mayor a 7, se aprueba.*
2. *Si no es el caso anterior, se prueba si la calificación promedio es por lo menos 6 y se tiene el crédito por participación.*
3. *En cualquier otra situación el alumno no aprueba el curso.*

Construya una proposición condicional de aprobación.

Veamos cómo podemos simbolizar las proposiciones del profesor.

1. a) p: $\frac{x+y}{2} > 7$
   b) s: *el alumno aprueba*

   La proposición condicional de aprobación es en este caso:

$$p \to s \tag{3.7}$$

2. a) p: $\frac{x+y}{2} > 7$
   b) q: $\frac{x+y}{2} \geq 6$
   c) r: $z = 1$
   d) s: *el alumno aprueba*

   La proposición condicional de aprobación es en este caso:

$$\neg p \land q \land r \to s \tag{3.8}$$

3. *Para determinar la proposición de aprobación definitiva, debemos combinar las premisas de las proposiciones 3.7 y 3.8 en una sola premisa π; haremos esto considerando un o inclusivo (¿por qué no usar una conjunción?):*

$$\pi := p \lor (\neg p \land q \land r)$$

$$\pi \to s \tag{3.9}$$

En efecto, el argumento lógico correcto y consistente es el que afirma que cuando π es verdadera, s **debe** ser verdadera, **si damos por hecho de que la implicación** $\pi \to s$ **es verdadera** (como en lluvia $\to$ nubes). Note que a la luz de la Observación 2.2, esto implica que si s es falsa, automáticamente π también es falsa ($\neg s \to \neg \pi$).

Lo anterior permite establecer entonces que, desde el punto de vista computacional (algorítmico), validar la condición suficiente (para el ejemplo π) es lo que permite determinar el flujo del procesamiento. Analicemos esto. Dado que se da por sentado que la implicación es verdadera, si la condición suficiente π es verdadera, sabemos que la condición necesaria s (aprobación del curso) también lo es. En caso contrario, es decir, si π es falsa, es porque s **lo es también**, en virtud de que la implicación es verdadera.

Observación 3.1. Note que para el Ejemplo 3.7 tenemos que el argumento:

$$\pi \vdash s$$

es válido, ya que la implicación es tautológica. Se deja la explicación de esta afirmación para el Ejercicio 3.7.10.

3.7. Ejercicios y Problemas

1. Coloque todos los paréntesis donde deban ir para que las siguientes proposiciones sean semánticamente claras y de conformidad con las convenciones de alcance. Si hay más de una posibilidad escríbalas todas.

   a) $\neg p \wedge q$

   b) $\neg p \vee q \wedge r$

   c) $p \to q \wedge r \wedge s$

   d) $p \vee q \to r \vee s$

   e) $\neg\neg p \wedge \neg q \to t \vee p$

   f) $p \to r \to s$

2. Escriba la tabla de verdad de las siguientes proposiciones lógicas

   a) $\neg p \wedge q$

   b) $\neg p \vee q$

   c) $p \wedge (q \vee r)$

3. Escriba la tabla de verdad de las siguientes proposiciones lógicas

a) $(p \to q) \land (r \to s)$

b) $p \to (q \to (r \to s))$

c) $p \land ((q \to r) \lor s)$

4. Diga cuál de las siguientes proposiciones es equivalente a $p \to (q \lor r)$.

   a) $q \lor (\neg p \lor r)$

   b) $q \land \neg r \to p$

   c) $p \land \neg r \to q$

   d) $\neg q \land \neg r \to \neg p$

5. Diga cuál de las siguientes proposiciones es equivalente a una tautología y cuál a una contradicción.

   a) $(p \land q) \to p$

   b) $\neg(\neg p \lor q) \land \neg(\neg q \lor r)$

   c) $(p \to q) \land \neg(q \to p)$

   d) $q \to (p \to (p \to (q \to p)))$

6. Suponga que $b < a$, $x = a + b$, $y = a - b$, y $z = a^2 - b^2$. Diga cual de las siguientes proposiciones es verdadera:

   a) $z < 0$

   b) $z < xy$

   c) $z = yx$

   d) $z \geq xy$

7. Diga en cuál de los siguientes silogismos el razonamiento es correcto y en cuál no. Discuta en clase las razones por las que un argumento es incorrecto.

   a) 1) Todos los automóviles tienen motor.
   2) Todos los motores utilizan aceite.
   3) Todos los automóviles utilizan aceite.

   b) 1) Todos los planetas del sistema solar giran alrededor del sol.
   2) La Luna es parte del sistema solar.
   3) La Luna gira alrededor del sol.

   c) 1) Algunos seres vivos tienen alas.
   2) Todos los mexicanos son seres vivos.
   3) Algunos mexicanos tienen alas.

8. Diga en cuál de los siguientes silogismos el razonamiento es correcto y en cuál no. Discuta en clase las razones por las que un argumento es incorrecto.

   a) 1) Ningún hombre puede volar.
   2) Los canarios vuelan.
   3) Ningún canario es un hombre.

   b) 1) Ninguna pesadilla es agradable.
   2) Las experiencias desagradables no se buscan.
   3) Nadie se acuesta buscando padecer una pesadilla.

   c) 1) Ningún fósil sufre un desengaño amoroso.
   2) Una ostra puede sufrir un desengaño amoroso.
   3) Las ostras no son fósiles.

   d) 1) El león cree que todos son de su condición.
   2) Ninguna oveja es un león.
   3) Ninguna oveja cree tener la condición de un león.

9. Muestre que el argumento del Ejemplo 3.5 es falso, mediante una tabla de verdad que permita comprobar que la Definición 3.5 no aplica a (3.4).

10. Explique, ayudándose de una tabla de verdad, por qué podemos afirmar que el argumento $\pi \vdash \eta$ es válido, cuando la implicación $\pi \to \eta$ es una tautología. Formule su explicación en español, escribiendo una oración del tipo "*Podemos afirmar que el argumento* $\pi \vdash \eta$ *es válido, cuando la implicación* $\pi \to \eta$ *es una tautología*, porque *si . . . entonces . . .*".

11. Diga cuáles de los siguientes argumentos son válidos y cuáles no ayudándose con tablas de verdad.

   a) $p \to (p \to q), p \vdash q$
   b) $\neg p \vee (q \to p) \vdash \neg p \wedge q$
   c) $\neg r \to (p \vee q), r \wedge \neg q \vdash r \to q$

12. Pedro sabe que tiene 2 veces más canicas que Juan, quien posee una canica menos que Diana, y también sabe que Ceci tiene 10 canicas más que Diana. Pedro sabe que la persona que le puede dar toda la información que necesita para saber cuántas canicas tiene cada quién, tiene entre 1 y 8 canicas. Para saber el número exacto de canicas que tiene esta persona, astutamente le pide que responda a 3 preguntas, diciendo sólo sí, o no. Si al término de las 3 preguntas, Pedro adivina cuantas canicas tiene esta persona, entonces se lleva todas las canicas.

   a) ¿A quién tiene que hacer las preguntas Pedro?

b) ¿Cómo es que Pedro podrá saber el número exacto de canicas que posee esta persona con sólo 3 preguntas?

c) Escriba las proposiciones lógicas que harían que Pedro adivinara el número 7.

d) ¿Cuantas canicas se lleva Pedro en este escenario?

13. Una compañía constructora busca un terreno adecuado para un nuevo proyecto. Los criterios de adecuación son los siguientes: Si el terreno está en pendiente o tiene menos de 1000m^2, no es adecuado; tampoco lo es si la superficie excede 2000m^2 o colinda con campos de cultivo; el uso de suelo debe ser comercial para que sea adecuado. Dadas las siguientes proposiciones:

p: El terreno está en pendiente.

q: tiene menos de 1000m^2.

r: tiene menos de 2000m^2.

s: colinda con campos de cultivo.

t: tiene uso de suelo comercial.

η : es adecuado.

Determine cuál de las siguientes opciones es la proposición que afirma que la empresa puede tomar la decisión de comprar el terreno.

a) $((p \wedge \neg q) \rightarrow r) \vee (s \rightarrow t)) \rightarrow \neg\eta$.

b) $(\neg p \wedge \neg q) \vee (r \wedge \neg s \wedge t) \rightarrow \eta$.

c) $\neg p \wedge \neg s \wedge t \wedge r \wedge \neg q \rightarrow \eta$.

d) $(r \vee \neg s \vee t) \wedge \neg(p \vee q) \rightarrow \eta$.

14. Se desea saber mediante un algoritmo si una cadena c de entrada contiene n caracteres. Suponga que el algoritmo lee un caracter mientras haya caracteres válidos e incrementa un contador si el caracter leído es válido. Considere las siguientes proposiciones:

p : El caracter leído es válido.

q : La longitud de la cadena es menor a n.

r : La lectura continúa.

a) Escriba la proposición lógica que describe la condición de paro de la lectura de la cadena. NOTA: Ponga atención a la ubicación correcta de los paréntesis.

15. Tiene una urna con 5 pelotas, 2 rojas y 3 azules. Usted no puede ver qué pelota extrae, y en consecuencia saca una pelota al azar. Usted tiene que sacar todas las pelotas rojas dentro de un máximo de 3 extracciones.

a) Escriba la proposición que afirma que logró su objetivo al cabo de la tercera extracción. NOTA: la semántica de la proposición debe reflejar el orden de extracción de las pelotas. ¡Cuidado a la ubicación de los paréntesis!

16. Un sistema de calefacción funciona en base a un principio de histéresis. El termostato, que controla el encendido de la calefacción, se activa cuando la temperatura del medio ambiente llega a 20°centígrados, y se paga cuando la temperatura apenas rebasa los 23°.

    a) Escriba la proposición que describe el estado de encendido del termostato.

    b) Escriba la proposición que describe el estado de apagado del termostato.

    c) Escriba la tabla de verdad que muestra que el termostato funciona correctamente.

    d) Utilizando la tabla de verdad del inciso anterior, escriba el argumento que muestra que el termostato funciona correctamente. Se le pide que escriba algo como $\pi_1, \pi_2, \ldots, \pi_n \vdash \eta$.

17. tiene 2 jarras de agua, una de un 3L., y otra de 4L. Usted sólo puede llenar cada jarra hasta su máxima capacidad, verter el contenido de una jarra en una segunda jarra hasta que ésta se llene (es posible que la primera tenga un sobrante si su capacidad excede a la capacidad de la segunda), o tirar al piso el contenido de una jarra por completo. Si una jarra está a su máxima capacidad, no se puede agregar más líquido. El punto de partida son ambas jarras vacías. Usted quiere medir exactamente 2 litros.

    a) Escriba las 6 proposiciones atómicas que declaran las 3 acciones posibles para cada jarra. Puede usar símbolos del tipo l_3 : "Lleno la jarra 3L."; v_3 : "Vierto el contenido de 3L en la jarra 4L.".

    b) Escriba las 9 proposiciones atómicas que declaran el estado de cada jarra. Puede usar símbolos del tipo c_{33} : "La jarra 3L contiene 3 litros (está llena)".

    c) Enumere los pasos de la estrategia más rápida para medir exactamente 2 litros, describiendo en castellano lo que cada paso representa. Por ejemplo, usted puede escribir:

    1) $l_3 \rightarrow c_{33}$ Si lleno 3L, la jarra contiene 3 litros.

    2) $\vdots$ ...

    El último paso de su estrategia debe afirmar que usted tiene 2 litros en alguna jarra.

    d) Escriba la proposición compuesta que declara la forma en que usted mide exactamente 2 litros. NOTA: la semántica de la proposición debe reflejar claramente el orden de las acciones. Coloque cuidadosamente los paréntesis para reflejar correctamente la semántica de la proposición.

Capítulo 4

Algoritmos

Bruno Lara, Jorge Hermosillo

4.1. Definición de un algoritmo

Definición 4.1. Un algoritmo es una secuencia ordenada de pasos para resolver un problema, de una determinada manera, mediante instrucciones definidas y finitas.

El ejemplo mas común, con el que todos nos hemos encontrado, es una receta de cocina o las instrucciones para cambiar la llanta ponchada de un automóvil. Como podemos imaginar, existen distintos grados o niveles de explicación, o abstracción, cuando hablamos de estos procedimientos.

Ejemplo 4.1

Secuencia de pasos para cambiar una llanta

1. *subir auto con el gato mecánico*
2. *quitar llanta ponchada*
3. *poner llanta nueva*
4. *bajar auto*

Ejemplo 4.2

Secuencia de pasos para cambiar una llanta

> 1. *abrir la cajuela del auto*
> 2. *extraer gato mecánico*
> 3. *extraer llanta de repuesto*
> 4. *extraer llave para extracción de birlos*
> 5. *asegurar que el auto no se mueva*
> 6. *aflojar todos los birlos, sin extraerlos completamente*
> 7. *subir auto con el gato mecánico*
> 8. *sacar los birlos*
> 9. *quitar llanta ponchada*
> 10. *poner llanta de repuesto*
> 11. *poner birlos*
> 12. *bajar auto*
> 13. *apretar birlos*
> 14. *guardar gato mecánico en la cajuela*
> 15. *guardar llave para extracción de birlos*
> 16. *cerrar la cajuela del auto*

Ejercicios:

- escribe el procedimiento para preparar una torta de jamón en 5 pasos
- escribe el procedimiento para preparar una torta de jamón en 10 pasos

Entre mayor sea el grado de abstracción en un procedimiento, menos pasos, y por ello, estos se convierten en los mas importantes. Es por esto que podemos decir que la mejor manera de comenzar a escribir un procedimiento o algoritmo es escribiendo los pasos mas importantes. Una vez hecho esto, podemos ver cuales de estos se pueden ir describiendo a mayor detalle.

Características de un algoritmo Los algoritmos en general deberán cumplir con las siguientes características:

- **finitos**: siempre terminan.
- **precisos**: la secuencia de acciones tiene un orden preciso.
- **definidos**: si se ejecuta dos veces el mismo algoritmo, con los mismos datos, se debe llegar a la misma solución.

Cada paso en un algoritmo deberá:

- realizar una tarea solamente
- acercarnos a la solución
- ser entendible
- no ser muy detallado

La tarea de lavarse los dientes por la mañana, puede describirse como una serie de pasos.

Ejercicio:

- Escribir los pasos necesarios para lavarse los dientes

Para escribir este algoritmo, nos hicimos preguntas implícitas, como por ejemplo: ¿tiene pasta el cepillo?, o ¿ya están limpios los dientes? ¿Qué otras preguntas pueden surgir?
Ejercicio:

- Escribe todas las preguntas que se te ocurran en relación a la tarea de lavarse los dientes.
- Al describir los pasos necesarios ¿cuantas acciones -verbos- están implicados?
- ¿Hay procesos o acciones que se repiten? si es así ¿que tanto se repiten?

Por lo general, las preguntas como las anteriores, se pueden responder con un simple "sí" o "no", verdadero o falso. Por lo tanto, podemos plantear algunas de estas preguntas como proposiciones lógicas:

- p : "El cepillo tiene pasta".

$$\text{¿tiene pasta el cepillo?} \Rightarrow \begin{cases} \text{si} & p \equiv V \\ \text{no} & p \equiv F \end{cases}$$

- q : "Los dientes están limpios".

$$\text{¿ya están limpios los dientes?} \Rightarrow \begin{cases} \text{si} & q \equiv V \\ \text{no} & q \equiv F \end{cases}$$

Durante la ejecución del algoritmo, las proposiciones son evaluadas y su resultado nos lleva por caminos diferentes para llegar al final de los pasos.

4.2. Análisis de problemas

Algunas veces, resolver un problema escribiendo la secuencia de pasos adecuada, o correcta, es una tarea sencilla, como es el caso de los problemas que hemos planteado hasta ahora: *cambiar una llanta*, *preparar una torta*, *lavarse los dientes*. Debido a que estamos familiarizados con estas tareas, nos resulta relativamente fácil *elaborar un plan* de acción para resolver el problema.

Sin embargo, no siempre tendremos frente a nosotros problemas sencillos u obvios de resolver, por lo que resulta útil contar con alguna estrategia o método de abordaje de un problema. En esta sección les proponemos uno muy sencillo, basado en una estrategia propuesta por el matemático George Polya.

Para Polya, todo intento por resolver un problema debe seguir el siguiente proceso:

1. Entiende el problema
2. Configura un plan
3. Ejecuta el plan
4. Revisa y extiende

Como vemos, lo primero es entender el problema. Para ello, proponemos aquí la siguiente estrategia:

Método 4.1 (Estrategia para entender un problema)**.** Hazte las siguientes preguntas:

1. ¿Cuál es el objetivo?
2. ¿Cuáles son los datos de entrada?
3. ¿Cuál es la salida deseada?
4. ¿Hay alguna condición?
5. ¿Es la condición suficiente para saber cómo procesar la entrada?
6. ¿Qué componentes tiene la tarea o el problema?

A través de estas preguntas es posible contextualizar el problema y entender su naturaleza. Estas preguntas podrían ser o no suficientes para resolver el problema. Pero no hay que olvidar que en esta etapa estamos en el qué, no en el cómo. Lo importante aquí es plantearse las preguntas que nos hagan sentido para entender qué es lo que tenemos que resolver. Para ello, otra estrategia es escribir el problema en nuestras palabras, o incluso hacer diagramas, dibujos o cualquier cosa que nos ayude a dar sentido al problema desde nuestra perspectiva. De esta forma, nos podremos dar cuenta de:

- Qué información tengo: lo que sí sé (especifica).
- Qué información me falta: lo que no sé (especifica).
- Qué debo lograr: ¿cómo sé que resolví el problema? (especifica).
- Qué cosas deben ocurrir y en qué orden: ¿cómo debería funcionar?

Veamos si podemos utilizar esta estrategia para resolver el problema de hacer la torta de jamón.

Ejemplo 4.3

Hacer una torta de jamón.

1. *¿Cuál es el objetivo?* ⇒ *Una torta de jamón.*
2. *¿Cuáles son los datos de entrada?* ⇒ *Los ingredientes.*
3. *¿Cuál es la salida deseada?* ⇒ *La torta armada: los ingredientes en su lugar.*
4. *¿Hay alguna condición?* ⇒ *Que lleve todo en forma adecuada.*

5. *¿Es la condición suficiente para saber cómo procesar la entrada? ⇒ Sí, si sabemos qué significa "forma adecuada". Aquí podemos recurrir a nuestras preguntas:*
   - *Qué información tengo: lo que sí sé (especifica): sabemos cómo es una torta.*
   - *Qué información me falta: lo que no sé (especifica): Supongamos que la queremos con queso, jitomate y lechuga.*
   - *Qué debo lograr: ¿cómo sé que resolví el problema? (especifica). La torta armada.*
   - *Qué cosas deben ocurrir y en qué orden: ¿cómo debería funcionar? Tener utensilios, cortar los ingredientes, armar la torta.*
6. *¿Qué componentes tiene la tarea o el problema? Los ingredientes, los utensilios.*

Primera aproximación algorítmica para resolver el problema:

1. *Elegir los ingredientes: pan, jamón, queso, jitomate y lechuga.*
2. *Elegir los utensilios: cuchillo.*
3. *Cortar el pan en dos mitades con el cuchillo.*
4. *Cortar el jamón, el queso y la lechuga con el cuchillo de manera que se adapten al tamaño del pan.*
5. *Cortar el jitomate en rodajas.*
6. *Colocar sobre una mitad del pan el jamón y el queso.*
7. *Colocar las rodajas de jitomate y la lechuga cortada sobre el queso y el jamón.*
8. *Colocar la otra mitad del pan sobre lo anterior.*

Veamos ahora otros ejemplos.

Ejemplo 4.4

Montar un obra de teatro. *Muchos hemos ido a ver una obra de teatro, pero seguramente pocos hemos montado alguna. Veamos si podemos elaborar un plan de acción.*

1. *¿Cuál es el objetivo? ⇒ Montar una obra de teatro.*
2. *¿Cuáles son los datos de entrada? ⇒ El guión de la obra.*
3. *¿Cuál es la salida deseada? ⇒ Actores teniendo un rol del guión en un escenario.*

4. *¿Hay alguna condición? ⇒ Tener el guión, los actores y el escenario adecuado.*
5. *¿Es la condición suficiente para saber cómo procesar la entrada? ⇒ Sí, si sabemos qué significa "escenario adecuado". Aquí podemos recurrir a nuestras preguntas:*
   - *Qué información tengo: lo que sí sé (especifica): sabemos cómo es una obra de teatro.*
   - *Qué información me falta: lo que no sé (especifica): La escenografía (iluminación, vestuario, mobiliario).*
   - *Qué debo lograr: ¿cómo sé que resolví el problema? (especifica). El guión puesto en escena.*
   - *Qué cosas deben ocurrir y en qué orden: ¿cómo debería funcionar? Tener los actores, tener al director, tener el escenario (teatro), tener la escenografía, ensayar la obra, promoverla, montarla.*
6. *¿Qué componentes tiene la tarea o el problema? Guión, actores, escenario.*

Primera aproximación algorítmica para resolver el problema:

1. *Elegir el guión de la obra.*
2. *Conseguir al director.*
3. *Conseguir los actores para cada rol.*
4. *Buscar un escenario donde se desarrollará la obra.*
5. *Conseguir todo lo necesario para la escenografía.*
6. *Ensayar la obra*
7. *Promover la obra.*
8. *Montar la obra.*

Ejemplo 4.5

Realizar una compra por internet. *En estos tiempos, seguramente hemos tenido que realizar compras por internet, pero ¿te has preguntado cómo resolver esta tarea mediante un algoritmo? Veamos si podemos elaborar un plan de acción.*

1. *¿Cuál es el objetivo? ⇒ Hacer una compra por internet.*
2. *¿Cuáles son los datos de entrada? ⇒ El producto a comprar, posiblemente sitios*

dónde buscar.

3. *¿Cuál es la salida deseada? ⇒ Una orden de compra.*
4. *¿Hay alguna condición? ⇒ Mejor precio para el producto deseado.*
5. *¿Es la condición suficiente para saber cómo procesar la entrada? ⇒ Sí, si sabemos qué características debe tener el producto. Aquí podemos recurrir a nuestras preguntas:*
   - *Qué información tengo: lo que sí sé (especifica): sabemos qué queremos.*
   - *Qué información me falta: lo que no sé (especifica): Las propiedades del producto (calidad, precio).*
   - *Qué debo lograr: ¿cómo sé que resolví el problema? (especifica). La orden de compra.*
   - *Qué cosas deben ocurrir y en qué orden: ¿cómo debería funcionar? Buscar sitios, buscar producto, comparar precios, elegir mejor opción, fincar pedido, pagar.*
6. *¿Qué componentes tiene la tarea o el problema? Producto, rango de precios, sitios de compra, método de pago, fechas de entrega.*

Primera aproximación algorítmica para resolver el problema:

1. *Describir el producto deseado.*
2. *Elegir puntos de venta de productos afines al que se desea.*
3. *Buscar el producto.*
4. *Comparar precios.*
5. *Elegir sitio de venta.*
6. *Fincar el pedido para el producto deseado.*
7. *Realizar el pago.*
8. *Guardar orden de compra.*

Ejercicios:
Ensaya el Método 4.1 para proponer algoritmos sencillos que resuelvan los siguientes problemas:

1. Dibujar en una computadora un plano 2D de una casa.
2. Mover un piano en una mudanza.
3. Armar un librero.
4. Resolver el problema 17 de la Sección 3.7.

4.3. Algoritmos en máquinas

Nuestro particular interés es en algoritmos que nos ayudan a solucionar un problema usando una computadora. En este sentido, podemos pensar en un algoritmo como un acercamiento a la solución de éste.

En términos del grado de abstracción, tenemos el enunciado o planteamiento del problema, en lo que podríamos denominar el nivel mas alto. Como hemos visto en la sección anterior, éste por lo general esta planteado en un lenguaje que es entendible y claro. El problema se analiza y una vez comprendido se recomienda elaborar los pasos principales, identificando las partes mas importantes. Después, se recomienda comenzar a escribir un algoritmo tomando en cuenta que la fase final consistirá en llevar nuestro algoritmo a algún lenguaje de programación que puede ser ejecutado en una computadora.

Estas máquinas, en general, tienen sólo cuatro habilidades:

- **lectura:** leer datos de un mecanismo externo a la memoria
- **escritura:** escribir datos de la memoria a un mecanismo externo
- **cálculo:** llevar a cabo simples operaciones sobre los valores en la memoria
- **control:** llevar control de lo anterior mediante secuencia, selección e iteración

Para resolver problemas por medio de computadoras, contamos con:

- condiciones de inicio, también conocidas como precondiciones
- condiciones finales, también conocidas como postcondiciones

Lo que nos lleva de las precondiciones a las postcondiciones, es precisamente un algoritmo. Entonces:

Algoritmo computacional: Conjunto de instrucciones que se ejecutan en una computadora de manera secuencial y que nos llevan de condiciones iniciales a condiciones finales. Esto es, dada una entrada (datos), una algoritmo nos entrega datos de salida, una solución.

Por lo general, cada uno de estos pasos o instrucciones son un cálculo, una operación que se lleva a cabo sobre los datos. Los datos, se alojan en entidades llamadas variables.

Definición 4.2 (Variable). A diferencia de las matemáticas, en donde una variable representa una cantidad que varía, en computación, una variable es un **contenedor de información** que la computadora mantiene en algún lugar físico de la memoria. Un algoritmo puede **escribir información** en el contenedor, lo que se conoce como **asignación a la variable**, y el contenido de ésta permanece guardado mientras no haya otra asignación. Cuando queremos recuperar el contenido del espacio físico asignado, se realiza una **lectura de la variable**, es decir se **lee el valor** del contenedor.

Como una variable es un contenedor para un dato, ésta tiene un valor que puede cambiar durante la ejecución de un algoritmo. Es común que cuando analizamos el planteamiento de un problema, los **sustantivos** más importantes sean nuestras **variables** más importantes. Así mismo, los **verbos** que encontramos en el planteamiento, por lo general se convierten en los **procesos** más importantes del algoritmo.

En general, los datos pueden ser de tres tipos:

- numéricos
- alfabéticos
- lógicos

Así mismo, contamos con cuatro tipo de operadores:

- asignación
- aritméticos
- relacionales
- lógicos

Es importante mencionar que estos operadores son estrictamente binarios, es decir requieren de dos y solo dos operandos para ser utilizados.

Operador de asignación
Es aquel operador que nos sirve para asignarle un valor a una variable:

Ejemplo 4.6

Asignación de valor a una variable, los operandos son el valor y la variable:

- $x = 4 \leftarrow$ *asigna el valor de* 4 *a la variable* x
- $y = x \leftarrow$ *asigna el valor de* x *a la variable* y
- $birloNo = 5 \leftarrow$ *asigna el valor de* 5 *a la variable* $birloNo$*, esto podria ser el número de birlos que sujetan a una llanta*

Operadores aritméticos
Son aquellos operadores mas sencillos que conocemos de las matemáticas:

- suma (+)
- resta (-)
- división (/), puede ser entera o real.
- multiplicación (*)
- modulo (%) - este operador regresa el residuo de una división.

Junto con el operador de asignación podemos comenzar a escribir cálculos:

Ejemplo 4.7

Asignación usando operadores aritméticos:

- $x = y + z \leftarrow$ *asigna el valor de la suma de las variables* y *y* z *a la variable* x*.*
- $x = 68 - 60 \leftarrow$ *asigna el valor de* 8 *a la variable* x*.*
- $x = 5 * 2 \leftarrow$ *asigna el valor de* 10 *a la variable* x*.*
- $x = 15\,\%5 \leftarrow$ *asigna el valor de* 0 *a la variable* x*.*
- $x = 17\,\%5 \leftarrow$ *asigna el valor de* 2 *a la variable* x*.*
- $x = 17/3 \leftarrow$ *asigna el valor de* 5.66666 *a la variable* x*.*
- $x = 18/2 \leftarrow$ *asigna el valor de* 9 *a la variable* x*.*

En estos casos, tenemos que para los operadores aritméticos, se requieren dos operandos, es decir dos cantidades o valores. Una vez realizada la operación, el resultado se convierte en el segundo operando para el operador de asignación. Cuando mas de un operador aritmético se encuentra en una expresión, ésta se evalúa en orden de importancia siguiendo la regla conocida como PEMDAS:

- **P**arentesis
- **E**xponienciación
- **M**ultiplicación
- **D**ivisión
- **A**dición
- **S**ustracción.

Cuando se encuentran dos operadores de la misma importancia en una expresión, éstos se evalúan de izquierda a derecha.

Ejemplo 4.8

Resultado de operaciones con operadores aritméticos de la misma importancia:

- $x = 3 + 2 * 4 - 5 \Rightarrow$ *El valor en* x *será:* $\{3 + [8]\} - 5 = 6.$
- $x = 6/2 * 3 + 6 \Rightarrow$ *El valor en* x *será:* $\{[3] * 3\} + 6 = 15.$
- $x = 7 - 5 * 4/2 + 1 \Rightarrow$ *El valor en* x *será:* $7 - \big\{\big([20]/2\big) + 1\big\} = -4.$
- $x = ((5 - 2) * 4) * 2/3 \Rightarrow$ *El valor en* x *será:* $\big[\big((3) * 4\big) * 2\big]/3 = 8.$

Operadores relacionales

Estos son los que nos permiten hacer comparaciones entre los valores de los datos o las variables, por lo general contamos con:

Relación	Símbolo
igual que	==
distinto que	!=
menor que	<
mayor que	>
menor o igual que	<=
mayor o igual que	>=

Tabla 4.1: Operadores relacionales

Observación 4.1. El resultado de una operación relacional es un valor lógico de *verdadero* o *falso*. De manera que si asignamos el resultado de alguna de estas operaciones, obtendremos valores de 1 (*verdadero*) o 0 (*falso*).

Ejemplo 4.9

Resultado de operaciones con operadores relacionales:

- $3 > 2 \Rightarrow$ *El resultado de esta operación es* 1 *(verdadero).*
- $3 != 5 \Rightarrow$ *El resultado de esta operación es* 1 *(verdadero).*
- $x = 3 > 5 \Rightarrow$ *El valor en* x *será:* 0 *(falso).*
- $8 == 4 + 4 \Rightarrow$ *El resultado de esta operación es* 1 *(verdadero).*
- $x = 5 == 7 \Rightarrow$ *El valor en* x *será:* 0 *(falso).*

Observación 4.2. Note de los ejemplos que podemos combinar operaciones aritméticas con comparaciones. También note la diferencia entre **asignar** un valor a una variable ($=$) y **comparar** la igualdad de dos valores o variables ($==$).

Operadores lógicos

También de naturaleza binaria, estos nos ayudan a combinar proposiciones evaluando su veracidad (ver Sección 1.3.1 en donde encontraremos sus tablas de verdad) en algoritmos computacionales usaremos y, o y no.

Consideremos ahora el siguiente problema:

En un cine, todas las personas con edad par deberán sentarse del lado derecho de la sala, las personas con edad impar deberán hacerlo del lado izquierdo.

Para este problema, las condiciones de inicio son todos los números positivos. Las condiciones finales, es que una vez que sepamos la edad de la persona, deberemos decirle en donde se debe sentar en la sala.

4.3.1. Ejercicios

Para cada uno de los siguientes problemas, escribe un algoritmo de 5 y un algoritmo de 15 pasos. Identificar los verbos y los sustantivos en cada uno de los planteamientos, así como en los algoritmos.

1. Describe los pasos para elaborar un sándwich.
2. Es el cumpleaños de tu ser mas querido, y le quieres sorprender con su desayuno favorito. Cuales son los pasos necesarios.

3. El próximo viernes, tienes examen de Biología, cuales son los pasos mas importantes para que te obtengas una buena calificación.
4. Mañana tienes clase presencial en la uni alas 9:00 a.m., que pasos necesitas llevar a cabo para llegar a tiempo.
5. Esta navidad te toca organizar los regalos, en tu grupo de la escuela quieren hacer un *amigo secreto*. ¿como lo lograrías exitosamente?

4.4. Operaciones de una computadora

Como hemos visto la computadora solo puede hacer 4 operaciones: leer, escribir, calcular, y controlar de que manera se llevan a cabo los pasos. Ahora, para poder controlar los pasos, solo hay tres opciones, secuencia, tomar decisiones y repetir pasos.

Secuencia
Los pasos de un algoritmo se ejecutan en un orden especifico. Así por ejemplo, tenemos la secuencia de pasos para el problema del cine:

Ejemplo 4.10

Secuencia de pasos para el problema del cine:

1. *dejar entrar a una persona*
2. *preguntar su edad*
3. *si la edad es par, indicar que deberá sentarse en la parte derecha de la sala*
4. *si la edad es impar, indicar que deberá sentarse en la parte izquierda de la sala*
5. *dejar entrar a la siguiente persona y regresar al paso 2*

Decisión
En algoritmos, tomar una decisión es equivalente a enrutar el procesamiento en función de una pregunta, cuya respuesta sólo puede ser *sí* o *no*, no existe punto medio. De esta forma, una vez que podemos responder a la pregunta, podemos enrutar el procesamiento realizando acciones pertinentes para cada caso.

Ejemplo 4.11

Para saber en dónde deberá sentarse una persona, la pregunta sería:

> *¿ Es la edad par?*
> ***si***
> *sentarse del lado derecho*
> ***no***
> *sentarse del lado izquierdo*

Cómo vimos en la Sección 4.1, podemos formular una pregunta en forma de una aserción (proposición), que en el lenguaje de la computación se le llama **condición**. De manera que responder a una pregunta, es equivalente a evaluar una condición, que es lo mismo que determinar el valor de verdad de la proposición equivalente.

En términos algorítmicos decimos que **evaluamos una condición**, es decir, evaluamos su valor de verdad: *sí* (proposición verdadera),*no* (proposición falsa). De esta forma, una vez que podemos evaluar la condición, podemos enrutar el procesamiento realizando acciones pertinentes para cada caso.

Para el ejemplo anterior, esto se traduce en escribir lo siguiente:

Ejemplo 4.12

> *Para saber en dónde deberá sentarse una persona, se evalúa la condición siguiente:*
>
> **si** *(edad es par)*
> *sentarse del lado derecho*
> **si no**
> *sentarse del lado izquierdo*

De este ejemplo, vemos que la respuesta a la pregunta, se traduce en **evaluar el valor de verdad de la condición *edad es par***, que sólo puede tener una respuesta válida a la vez, o la edad es par (condición **VERDADERA**) o no lo es (condición **FALSA**).

Las dos respuestas constituyen el *todo* de un universo, en este caso, los números enteros positivos, representados por el conjunto A en la fig. 4.1. Los números pares serán los que están en el conjunto B y los impares en el conjunto C.

Ahora, estas decisiones se pueden anidar, imaginemos que el problema cambia un poco de tal forma que ahora se lee así:

> En un cine, todas las personas con edad par deberán sentarse del lado derecho de la sala, las personas con edad impar deberán hacerlo del lado izquierdo. Las personas cuyas edades sean pares y además múltiplo de 5 deberán sentarse en las primeras filas.

Los pasos a seguir cambian ahora a:

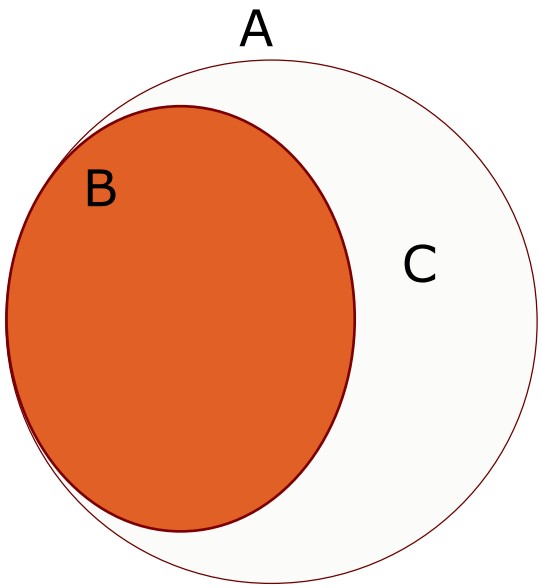


Figura 4.1: Universo de números enteros positivos, donde los pares pertenecen al conjunto B.

Ejemplo 4.13

Para saber en donde debera sentarse una persona, la pregunta sería:

¿ Es la edad par?
si
dirigirse al lado derecho
¿la edad de la persona es múltiplo de 5?
si
sentarse en las filas frontales
no
sentarse en las filas posteriores

no
sentarse del lado izquierdo

En términos de evaluar condiciones, este ejemplo se escribiría así:

Ejemplo 4.14

Condiciones anidadas para el ejemplo anterior:

si *(edad es par)*

dirigirse al lado derecho
si *(edad es múltiplo de 5)*
sentarse en las filas frontales
si no
sentarse en las filas posteriores
si no
sentarse del lado izquierdo

Como podemos imaginar, nuestro conjunto A de números enteros positivos cambia para incluir dentro de los números pares B a un subconjunto de aquellos números que son pares, pero que también son múltiplos de 5 (D), como se ve en la fig. 4.2

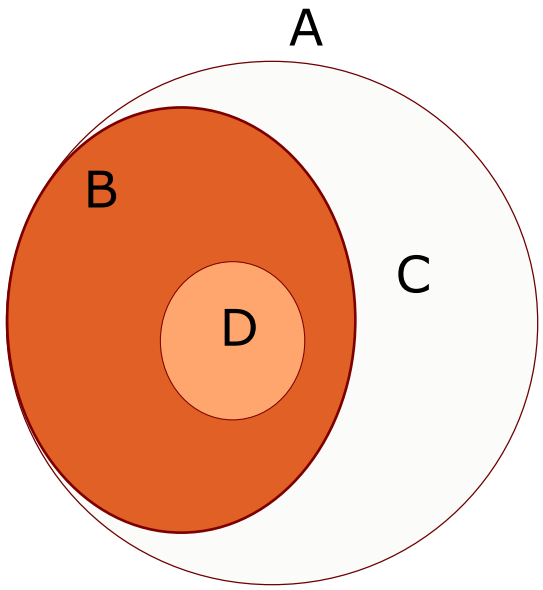


Figura 4.2: Números enteros y subconjuntos.

Para resumir lo anterior, lo mas importante para recordar, es que cuando escribimos algoritmos computacionales, cada vez que evaluamos la veracidad de una proposición lógica -derivada de una pregunta- solo hay dos repuestas y estas dos respuestas deberán cubrir el universo completo de soluciones. Esto nos permite darle un flujo a la secuencia de pasos necesarios para resolver nuestro problema.

Iteración
Esta estructura de control nos permite repetir procesos un número de veces. Generalmente, un proceso se repite hasta que se cumple alguna condición y generalmente son de dos tipos. En el primero, operaciones y/o procesos se ejecutan una vez, al final, se revisa si se ha cumplido alguna condición, de ser así, se detiene la ejecución y se continua con los pasos

siguientes, posteriores al ciclo de repetición.

Ejemplo 4.15

proceso A
repetir
proceso B
proceso B
hasta que *proposición sea verdadera*
proceso D

En este tipo de iteración, se ejecuta el proceso A, después se ejecutan los pasos B y C, se pregunta si la proposición es VERDADERA, en caso de que así sea, se continua con el proceso D. Cuando la proposición es FALSA, se volverá a ejecutar el proceso B y C, hasta que la proposición sea VERDADERA.

Ejemplo 4.16

Iteraciones de este tipo son:

- *vertir agua en la taza* ***hasta*** *que esta este llena*
- *girar el tornillo* ***hasta*** *que este quede apretado*
- *cepillar dientes* ***hasta*** *que queden limpios*
- *dejar agua en el fuego* ***hasta*** *que hierva*

En el segundo tipo de iteración, primero se evalúa si una proposición es VERDADERA, en caso de que así sea se ejecutan ciertos procesos y/o operaciones. Una vez ejecutados, se vuelve a evaluar la proposición. Los procesos se repetirán mientras no cambie el valor de verdad de la evaluación.

Ejemplo 4.17

proceso A
mientras *proposición sea verdadera* ***repetir***
proceso B
proceso C
proceso D

En este caso se ejecuta el proceso A, se evalúa la proposición, en caso de que esta sea VERDADERA se ejecutan los procesos B y C, se vuelve a evaluar la proposición, en caso

de que esta sea VERDADERA se repetirán los procesos B y C, en caso de que sea FALSA, el algoritmo continúa con la ejecución del proceso D, inmediatamente posterior al ciclo de repetición.

Ejemplo 4.18

Iteraciones de este tipo son:

- ***mientras*** *la taza no este llena* ***repetir*** *vertir agua*
- ***mientras*** *el tornillo no este apretado* ***repetir*** *girar*
- ***mientras*** *los dientes no esten limpios* ***repetir*** *cepillar*
- ***mientras*** *el agua no hierva* ***repetir*** *dejar en el fuego*

Para ambos casos la condición de paro es que la proposición sea verdadera. Como habíamos visto, en algoritmos computacionales, esta condición o proposición viene de una pregunta.

- ¿esta llena la taza?
- ¿esta apretado el tornillo?
- ¿estan limpios los dientes?
- ¿hierve el agua?

Así mismo, la respuesta solo tiene un valor valido **FALSO** o **VERDADERO**.

Aunque no existe una receta o formula exacta para elaborar un algoritmo, aquí mencionamos algunas recomendaciones.

- Analizar y comprender el enunciado del problema usando el Método 4.1.
- Identificar los procesos más importante observando los verbos.
- Identificar las decisiones mas importantes analizando las condiciones.
- Identificar los ciclos o repeticiones
- Identificar las variables observando los sustantivos.

En seguida:

- desarrollar un algoritmo de alto nivel, con pasos no muy detallados, intentando cubrir los procesos mas importantes
- hacer prueba de escritorio...si muestra problemas, resolverlos

Finalmente:

- incorporación de refinamientos, elaborar pasos con mas detalle
- agrupar procesos, cuando sea necesario
- agrupar variables, cuando sea necesario
- probar el algoritmo

Ahora, tomando en cuenta el problema del cine, junto con la Tabla 4.1 y las observaciones 4.1 y 4.2, un algoritmo correcto podría ser el siguiente:

En un cine, todas las personas con edad par deberán sentarse del lado derecho de la sala, las personas con edad impar deberán hacerlo del lado izquierdo. Las personas cuyas edades sean pares y además múltiplo de 5 deberán sentarse en las primeras filas.

Ejemplo 4.19

Algoritmo solución

comienzo
repetir
    dejar entrar al primer invitado
    preguntar su edad
    guardar su edad en la variable $edad$
    si $(edad\,\%2 == 0)$
        indicar que se dirija al lado derecho
        si $(edad\,\%5 == 0)$
            indicar que se siente en las hileras frontales
        si no
            indicar que se siente en las hileras traseras
    si no
    indicar que se siente del lado izquierdo
hasta que *no haya mas invitados*
fin

El algoritmo indica dejar entrar al primer invitado, se le pregunta su edad y esta se guarda en la variable $edad$. Se evalúa lo que está entre paréntesis, el resultado de aplicar el operador modulo $\%2$ a cualquier número positivo, será 0 si éste es par, por lo que la evaluación de esta proposición será VERDADERO, indicando al invitado a dirigirse al lado derecho. Ahora, el resultado de aplicar el operador modulo $\%5$ a cualquier número, será 0 si el número es un múltiplo de 5, por lo que la proposición evaluará VERDADERO e indicará al invitado a sentarse en las hileras frontales, de otra manera le indicará sentarse en las hileras posteriores. Cuando la primera evaluación es FALSA, el algoritmo indica al invitado sentarse del lado izquierdo. Este proceso se repite hasta que ***no*** *haya mas invitados* sea VERDADERO; en otras palabras, cuando *haya mas invitados* sea FALSO, el ciclo se romperá y el algoritmo terminará.

4.4.1. Ejercicios

Con las herramientas aprendidas hasta este momento, elabora los siguientes algoritmos con al menos 10 pasos.

1. Hacer un algoritmo que dados tres números enteros, evalúe si alguno de los números leídos es la suma de los otros dos.

2. Hacer un algoritmo que imprima si un número entero, leído por teclado, es positivo, negativo o cero.

3. Una balanza se encuentra en equilibrio cuando el producto de la carga aplicada sobre el brazo derecho por la longitud de este brazo es igual al producto de la carga aplicada al brazo izquierdo por la longitud de este otro brazo. Hacer un algoritmo, para determinar, dado cualquier conjunto de valores para las longitudes de los brazos y las cargas aplicadas, si la balanza está en equilibrio.

4. Escribir un algoritmo que, dadas 5 calificaciones (entre 0 y 100), calcule la calificación promedio y muestre la calificación final de acuerdo a la siguiente tabla:

promedio	calificación final
90-100	A
80-89	B
70-79	C
60-69	D
0-59	E

5. Escribir una algoritmo que lea una cantidad de grados Celsius e imprima su equivalente en grados Farenheit

4.5. Algoritmos abstractos

En esta sección estudiaremos los pasos necesarios para ir desde el enunciado del problema hasta el algoritmo mas abstracto, usando las herramientas que hemos estudiado. Este último algoritmo, se encuentra en un lenguaje muy cercano a un lenguaje computacional. Añadiremos dos herramientas mas, que son muy útiles para escribir algoritmos y que se han mencionado como operaciones que son capaces de llevar a cabo las computadoras.

Lectura
Un proceso importante para un algoritmo, es leer datos de algún mecanismo, este puede ser el teclado, algún archivo o la memoria, entre otros. Para fines de este curso usaremos la siguiente forma para asignar un valor que se pide al usuario y este introduce por medio del teclado a una variable.

Ejemplo 4.20

Sintaxis para lectura
nombre=entrada (cual es tu nombre:)
edad=entrada (cual es tu edad:)

En el primer ejemplo, al usuario se le pide introducir su nombre, este valor, en este caso un tipo de dato alfabético, sera guardado en la variable $nombre$. En el segundo ejemplo, al usuario se le pide introducir un tipo de dato numérico, que sera guardado en la variable $edad$. Podemos pensar en la palabra $entrada$ como un procedimiento que primero, imprime en la pantalla la frase entre paréntesis y después guarda lo que entre desde el teclado, en la variable que le precede al procedimiento.

Escritura
Normalmente, una vez que nuestro algoritmo ha terminado, es necesario que el algoritmo nos muestre algún resultado. Tradicionalmente, las maquinas *imprimían* los resultados en papel, por lo que se sigue usando la palabra *imprimir* para indicar el proceso de escritura. En nuestro caso, asumimos que nuestro mecanismo de impresión o de escritura es la pantalla, esto es, los resultados son mostrados en la pantalla. Siguiendo la misma linea, para fines de este curso usaremos la siguiente sintaxis.

Ejemplo 4.21

Sintaxis para escritura
imprimir (el promedio de las edades fue de:) promedio
imprimir (el resultado de la división de a entre b es de:) resultado

En el primer ejemplo, el algoritmo imprime la frase entre paréntesis *"el promedio de las edades fue de:"* así como el valor que guarda la variable *promedio.* en el segundo caso,

la frase *"el resultado de la división de a entre b es de:"* y el valor que esta guardado en la variable *resultado*.

Ahora podemos proceder a desarrollar un algoritmo de una manera mas precisa. Comenzaremos con el problema del cine.

Ejemplo 4.22

Algoritmo solución

comienzo
repetir
    edad= entrada (edad del invitado)
    si $(edad\,\%2 == 0)$
        imprimir (dirigirse al lado derecho)
        si $(edad\,\%5 == 0)$
            imprimir (sentarse en las hileras frontales)
        sino $(edad\,\%5 == 0)$
            imprimir (sentarse en las hileras posteriores)
    si no
    imprimir (sentarse del lado izquierdo)
hasta que *no hay invitados*
fin

Mientras siga habiendo invitados la evaluación de la proposición *no hay invitados* sera FALSO y se repetirá la ejecución de nuestro algoritmo.

Veamos ahora un nuevo problema usando **decisión**:

Una cierta compañía fabricante de repuestos de automóviles ha descubierto defectos en algunos de sus productos, específicamente aquellos cuyos números de serie se encuentran dentro de los rangos: 14,681 a 15,681, 70001 a 79,999 y 88,888 a 111,111. Escribir un algoritmo que permita, dado un número de serie, deducir si es o no defectuoso.

Comenzamos por identificar la variable mas importante y la única que necesitaremos en este ejemplo, eso es el número de serie de la pieza a evaluar, la llamaremos $noSerie$. Existen dos tipos de piezas, las defectuosas y las no defectuosas. Si lo vemos como conjuntos, entonces tenemos el conjunto de todas las auto piezas (conjunto **A**) y dentro de este, encontramos 3 conjuntos con las piezas defectuosas: el conjunto **B** con las piezas que van del número de serie de 14,681 a 15,681, el conjunto **C** con las piezas con número de serie de 70,001 a 79,999 y el conjunto **D**, cuyos número de serie van del 88,888 a 111,111. Eso se puede ver en la fig. 4.3

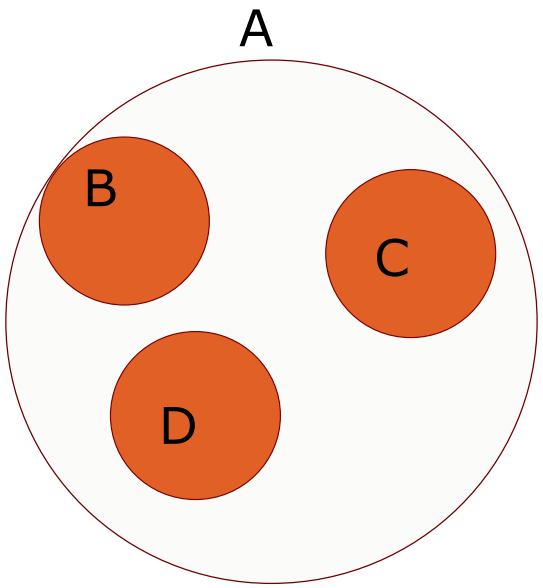


Figura 4.3: Conjunto de todas las auto piezas y subconjuntos de las piezas defectuosas.

Lo que nos interesa saber es si el número de serie ingresado esta en alguno de los conjuntos B, C ó D. '?qué piezas componen estos conjuntos? una primera aproximación a la resolución de este problema seria:

Ejemplo 4.23

Primera aproximación
comienzo
noSerie= entrada(teclea el número de serie de la pieza)
    si** la pieza esta en el conjunto **B ó** en el conjunto **C ó** en el conjunto **D
        imprimir (la pieza es defectuosa)
    si no
        imprimir (la pieza no es defectuosa)
fin

Podemos identificar ahora tres preguntas importantes:

1. ¿esta la pieza en el conjunto **B**?
2. ¿esta la pieza en el conjunto **C**?
3. ¿esta la pieza en el conjunto **D**?

¿Cuál es la condición para que una pieza se encuentre en el conjunto **B** si su número de serie es mayor o igual a 14,681 y menor o igual a 15,681? Formalizando esto tenemos que:

Ejemplo 4.24

*Condición para que una pieza este en el subconjunto **B***

comienzo si $((noSerie >= 14,681)$ *y* $(noSerie <= 15,681))$
    *imprimir (pieza en subconjunto **B**)*
si no
    *imprimir (pieza no esta en subconjunto **B**)*
fin

Es importante tomar en cuenta que en realidad hay dos proposiciones que se están evaluando, primero $(noSerie >= 14,681)$, esta tendrá un valor FALSO o VERDADERO, después tenemos $(noSerie <= 15,681)$ que de igual manera tendrá que ser evaluada. Como están unidas por un y, esto es una conjunción, ambas necesitan ser verdaderas para que se cumpla la condición, indicándonos que la pieza esta en el subconjunto **B**. Lo mismo sucede para los subconjuntos **C** y **D**.

Como habíamos dicho, un número de serie es defectuoso, si se encuentra en el subconjunto **B** o en el subconjunto **C** o en el subconjunto **D**. Esto es un **o**, una disyunción, por lo que bastara con que una de ellas evalué a VERDADERO para que la pieza sea defectuosa. Así, nuestro algoritmo podría quedar de la siguiente manera:

Ejemplo 4.25

Solución al problema de las piezas defectuosas:

comienzo
$noSerie$*=entrada (teclear número de serie de la pieza:)*
si $((noSerie >= 14,681)$ *y* $(noSerie <= 15,681))$
        o$((noSerie >= 70,001)$ *y* $(noSerie <= 79,999))$
        o$((noSerie >= 88,888)$ *y* $(noSerie <= 111,111))$
    imprimir (pieza defectuosa)
si no
    imprimir (pieza no defectuosa)
fin

Veamos ahora un nuevo problema usando **decisión** e **iteración**:

Escribir un algoritmo que pida al usuario un número positivo. El algoritmo deberá imprimir todos los números pares desde 0 hasta el número tecleado.

Una primera aproximación para resolver nuestro problema podría ser:

Ejemplo 4.26

Primera propuesta:

1. *pedir número a usuario*
2. *comenzar en 1*
3. *verificar si el numero es par*
4. *si el número es par imprimirlo*
5. *si no es par, no imprimir nada*
6. *pasar al siguiente número*
7. *regresar al paso 3*

Aquí, podemos identificar fácilmente nuestra primer variable, el número que el usuario tecleara, que llamaremos $noUsuario$. La segunda variable, es un contador, que debera recorrer desde 1 hasta el número $noUsuario$, por convención se usa la variable i.

Cada vez que repetimos un número de pasos, a esta repetición se le conoce como ciclo. De esta manera, nuestro ciclo se repetirá desde 1 hasta $noUsuario$, esto es $noUsuario$ veces.

Ejemplo 4.27

Solución al problema de los números pares:

comienzo
$noUsuario$=*entrada (teclear número maximo:)*
$i = 1$
mientras $(i <= noUsuario)$ ***repetir***
    si $(i\,\%2 == 0)$
        imprimir (numero par:) i
    $i = i + 1$
fin

Como podemos ver, al principio de nuestro algoritmo $i = 1$. La primera vez que se recorre el ciclo, se revisa primero que i sea menor que el número introducido por el usuario. Cuando es menor se entra al ciclo, se revisa que el i sea par, si lo es se imprime 'numero par'. En este caso en especial, no hay nada para cuando el valor de i es impar, el algoritmo

simplemente no hace nada. En cualquiera de los dos casos, i es par o impar, el algoritmo procede a incrementar a i una unidad ($i = i + 1$) y se regresa a evaluar la expresión **mientras**.

4.5.1. Ejercicios

Con los últimos conceptos adquiridos, escribir algoritmos mas precisos y abstractos para los siguientes problemas. Es importante mencionar que no es necesario ir directamente a una escritura abstracta, se recomienda primero hacer algoritmos de alto nivel, que nos permitan encontrar las variables, decisiones, ciclos y procesos mas importantes usando lenguaje de alto nivel. En un segundo paso, se puede refinar los pasos para obtener lo que podemos llamar un algoritmo de bajo nivel (ver Ej. 4.27 y 4.25).

1. se leen de teclado dos fechas en el formato día (1 a 31), mes (1 a 12) y año (entero de cuatro dígitos), correspondientes a la fecha de nacimiento y a la fecha actual, respectivamente. Escribir un programa que calcule y visualice la edad del individuo. Si es la fecha de un bebé (menos de un año de edad), se debe de dar en meses y días; en caso contrario, la edad se calculará en años.

2. hacer un algoritmo que sume los primeros 20 números naturales (uno por uno) y que imprima en cada paso las sumas que den un resultado par.

3. escriba un algoritmo que calcule e imprima la calificación final de N (introducida por el usuario al principio del algoritmo) alumnos. La calificación por cada alumno se genera de la siguiente manera:

   - 40 % de la calificación corresponde al promedio de 3 exámenes parciales
   - 15 % de la calificación corresponde al promedio de 4 tareas
   - 45 % de la calificación corresponde al examen final.

   Para cada alumno leer las calificaciones correspondientes a los exámenes parciales, las tareas y el examen final y asignarle una calificación representada por una letra, basándose en la siguiente tabla:

A	de 8.0 a 10
B	de 7.0 a 7.9
C	de 6.0 a 6.9
NA	de 0 a 5.9

4. escribir un algoritmo que pida las coordenadas de dos puntos A y B en el plano, escribir un algorimto que imprima la distancia euclideana entre ellos.

5. escribir un algoritmo que pida al usuario el radio de un circulo e imprima como resultado su área y su diámetro.