Contents

[Tabla de Conversion 2](#_Toc207491425)

[UNIFICACION DE UNA VARIBLE 3](#_Toc207491426)

[**Recuerde** 3](#_Toc207491427)

[MODUS PONENS 4](#_Toc207491428)

[SHORT – CUT (Corto Circuito) 4](#_Toc207491429)

[EL MODUS TOLLENS 5](#_Toc207491430)

[Usaremos el MT Algebraicamente (con letritas) 5](#_Toc207491431)

[Disparo(shoot) de una regla 5](#_Toc207491432)

[Silogismo disyuntivo (SD) 5](#_Toc207491433)

[Metodo Rapido (MT + SD) 6](#_Toc207491434)

[MT usando Factores Boole 6](#_Toc207491435)

[Negacion de los OPPREL’s (operadores relacionales) 7](#_Toc207491436)

[Algoritmos de encadenamiento (chaining algoritm’s) 7](#_Toc207491437)

[Forward Chainind (FWC) 7](#_Toc207491438)

# Tabla de Conversion

* Si el v esta en la premisa

If AvB if A

Then then

C = C

If B

Then

C

* Si el ^ esta en la conclusion

If P if P

Then then

A^B = A

If P

Then

B

Si el ^ esta en la conclusion

If P if P

Then then

A(); = A();

B(); If P

Then

B();

* Si el v esta en la conclusión

If P

Then

AvB

Una regla

If P

Then

Q

Expresa una implicación tautológica: P ⇒ Q

/\*

p=”estudio” q=”apruebo”

p →q = si estudio entonces apruebo

La implicación puede ser true o false

P = me saco 51 q=aprubo la materia

p ⇒q = si me saco 51 entonces apruebo la materia

La implicación es tautológica (siempre true)

\*/

Entonces la regla

If P

Then

AvB

Sera: p ⇒(AvB)

Escribiendo sin la doble línea

p →(AvB) = // ∝→β = ¬ ∝vβ

=¬pv(AvB)

=¬pvAvB

Nota. Cuando los conectivos son los mismos se quita los paréntesis, cason contrario no, se usa la distribución.

=¬(p^¬A)vB= // ¬ ∝vβ =∝→β

=(p^¬A) →B

If p^¬A

Then

B

Pero también se puedo haber hecho esto

¬pvAvB= Conmutativa

=¬pvBvA Ley de morgan

=¬(p^¬B)vA // ¬ ∝vβ =∝→β

=(p^¬B) →A

If p^¬B

Then

A

**Resumiendo.**

If P if P ^ ¬A

Then then

AvB = B

If P ^ ¬B

Then

A

If P if P ^ ¬B^ ¬C

Then then

AvBvC A

If P ^ ¬A^¬C

Then

B

If P ^ ¬A^¬B

Then

C

(.) Convertir a reglas esenciales las siguientes regla R

R) if (¬p→q)^s

Then

A→¬B

Solucion:

Lo que hace es convertir la premisa a FND y la conclusión a FNC, para luego aplicar la tabla de conversión (usando el arbol)

Premisa

=(¬p→q)^s ∝→β = ¬ ∝vβ

=(pvq)^s Distributividad

=(p^s)v(q^s) FND de 2 minterm

Conclusion

= A→¬B ∝→β = ¬ ∝vβ

= ¬AvB

La regla R queda

R) if (p^s)v(q^s)

Then

¬AvB

**R1 R2 R3 R4**

R = {R1, R2, R3, R4}

# UNIFICACION DE UNA VARIBLE

La asignacion en un lenguaje procedimental

X = 55

X = 80 //x=~~55~~ 80

La asignación puede cambiar su valor con la asignación (=)

Cuando la variable se unifica con un valor, la variable **es** ese valor

xU = 80 //xU = x unificada a 80

Ahora x ya no puede unificarse a otro valor

xU = 90 //ERROR

Un hecho sintácticamente tiene la forma de una conclusión de regla: var=valor

La Base de hechos (BH) es una lista de hechos, por ejemplo:

BH = {x=30, Edad=25, Presion=Alta}

(La base de hecho no tiene hechos repetidos)

La BH también nos dice:

* Que variables están unificadas, e.g. x, Edad, Presion.
* ¿Qué variables no están unificadas?

**Recuerde.**

* Si una variable esta en la BH, la variable esta unificada.
* Si una variable no esta en la BH, la variable NO esta unificada y vale null.

BH = [p=alta, z=0]

BH.add(temp=baja) //lo inserta sin problemas

BH = [p=alta, z=0, temp=baja]

BH.add(z=0) //ya esta en la BH, no pasa nada

BH.add(p=media) //no se puede puede modificar(contrdiccion), halt=Error=Exception

BH = [p=alta, z=0, temp=baja]

# MODUS PONENS

p →q true //si me traes chocoflan entonces voy al cine

p . true

q

if p

then BH = [., ., …]

va=valor

**True true**

**R1**) if (nubes ≠ estratos ^ x<20)

Then

Presion=baja ⇒ Presion=baja

BH = [nubes=cirros, x=10]

**R2)** if ¬p^q^s

Then

¬m

BH = [s, t, q, ¬p]

**R3)** if ¬p^q^s

Then //no hay **q**

q

BH = [s, t, r, ¬p]

**R4)** if ¬p^q^m

Then

z

BH = [s, t, q, ¬m] //no hay m ≠¬m

# SHORT – CUT (Corto Circuito)

v [ | | | ] v.length

0 1 2 3

I=0;

While(I < v.length && v[i] != 0) {

I++;

}

~~While(v[i] != 0 && i < v.length) {~~ //esto es un ERROR

~~I++;~~

~~}~~

El MP es un SE, testea la premisa sin usar short cut.

Si alguna variable de la premisa no esta unificada, no se testea el MP

**Ejemplo:**

**R) if Q>0 ^ P=a**  //P no esta unificada, NI siquiera testeo

Then //R⇒MP

M=si

BH = [Q=20, s=a, ..]

**true**

**false**

**R) if p<0 ^ z<>b** //la premisa no se valida (no es true)

Then //R⇒MP

M=si

BH = [p=4, z=c, ..]

Como las varibles de la premisa (p y q) estan unificadas, testeo el MP

El que mas confunde es el “v”

**R) if Q<>0 v z<>0 v P=a** //z no esta unificada, NO TESTEO el MP

Then //R⇒MP

Tiempo=frio

BH = [Q=8, P=a, ..]

# EL MODUS TOLLENS

Si la BH niega la conclusión, entonces se concluye la negación de la premisa.

If P

Then R MT ⇒ ¬P

Q

BH = [¬Q, …]

P →Q // P →Q ≡ ¬Q → ¬P

BH: P . (MP)

Q

P →Q ¬Q → ¬P

BH: ¬Q . BH: ¬Q .

¬p ¬p

# Usaremos el MT Algebraicamente (con letritas)

(.) Realizar el MT con la regla R y la BH dada.

R) if ¬A

Then

¬P // Antes de hacer el MT, verifique que la conclusión este negada x por un hecho

BH = [P, …]

Hay MT

RMT⇒A //¬¬A=A

**Pero el MT tiene un problema**

(.) Realizar el MT entre la regla R y la BH dada.

R) if P^¬Q

Then

¬S

BH = [S, …]

R MT⇒ ¬( P^¬Q) ≡ ¬PvQ

Pero ¬PvQ no puede insertarse (add) a la BH

/\*

Recordemos que la BH solo tiene literales (letra o negación de letra)

BH = [p, q, s, ¬r, ...]

\*/

# Disparo(shoot) de una regla

Se dice que una regla R dispara una conclusión c, anotado: R⇒c

Si lo arrojado por la regla R, puede insertarse a la BH

En el ejemplo anterior, se hizo el MT, pero lo obtenido (¬PvQ) no puede insertarse a la BH.

∴ R MT⇒

R) if ¬s ^ t ^ ¬q

Then

p

BH = [¬p, …]

R MT⇒ ¬ (¬s ^ t ^ ¬q) ≡ s v ¬t v q

(.)

R) if ¬p

Then

w

BH = [¬w, …]

R MT⇒ ¬ (¬p) ≡ p //esto lo puede add a la BH

# Silogismo disyuntivo (SD)

p v q p v ¬s v t q v ¬~~s~~ v p

BH: ¬p . BH: s, ¬t . BH: q, ~~s~~ .

q p q v p

**Solucion:**

**R⇒**¬(q ^ ¬s ^ t) ≡ ¬q v s v ¬t

Usando Silogismo Disyuntivo

¬q v s v ¬t

BH: s, q, t .

s //Esto se add a la BH

(.) Haga el MT con

R) if ¬A ^ P ^ Q

Then

w

BH: [¬w, ¬P, ¬A]

**Solucion**

R⇒ (¬A ^ P ^ Q) ≡ A v ¬P v ¬Q

BH: [¬w, ¬P, ¬A] .

¬P, ¬Q //Esto no puede add a la BH

∴ R MT⇒

# Metodo Rapido (MT + SD)

//Solo funciona para reglas simples

/\*

If minterm

Then

c

\*/

**Ejemplos:**

(.) Usando el MT rapido

R) if ¬A ^ P ^ s ^ w **Solucion:**

Then Hemos validado todos los literales de la premisa, EXCEPTO UNO (p), son validas.

C ∴ Concluyo la negacion del que falto validar: R MT⇒¬p

BH = [¬c, ¬A, s, w]

(.) Usando el MT rapido

R) if ¬p ^ q ^ ¬s **Solucion:**

Then Para aplicar el MT, SOLO UNO debe quedarse sin validar, pero aquí hay 2 (¬p y ¬s)

¬s ∴ **R MT⇒**

BH = [¬s, q]

(.) Si se validan todos

R) if ¬p ^ ¬q ^ ¬s **Solucion:**

Then Todos los literales de la premisa han sido validados (No hay ninguno sin validar)

¬r ∴ **R MT⇒**

BH = [r, ¬p, ¬s, ¬q]

# MT usando Factores Boole

(Utilizamos el MT)

If p Solucion

Then No se puede colocar x ≠ 3, debe ser: var = valor

x=3

BH = [x<>3, ...]

*¿Cuando hay contradiccion entre una conclusión de una regla y un hecho?*

Hay contradicción, cuando la conclusión y un hecho tienen la misma variable, pero diferentes valores.

Ejemplo:

R) if ¬p **Solucion:**

Then var1 ≠ var2

Var = valor1

BH = [var = valor2]

# Negacion de los OPPREL’s (operadores relacionales)

¬(A≠B) ≡ A=B ¬(A<B) ≡ A≥B ¬(A≥B) ≡ A<B

¬(A=B) ≡ A≠B ¬(A>B) ≡ A≤B ¬(A≤B) ≡ A>B

(.) Usando el MT, haga el MT con la regla R y la BH dada

R) if p>0 ^ R<>a ^ z<>10 **Solucion**

Then El literal (Factor Boole), R<>a no se valida con la BH

Q=b R MT⇒ ¬(R≠a) ≡ R=a

BH = [Q=c, p=20, z=5] **∴ R MT⟾ R=a**

Sin embargo, muchas veces el MT puede arrojar factores Boole, que no pueden add a la BH

Por ejemplo:

R) if Q>0 ^ P=alta ^ Temp>100 **Solucion**

Then Usando el MT

x=20 R MT⇒ ¬(Temp>100) ≡ Temp≤100, esto no es un hecho

BH = [x=10, Q=50, P=alta] ∴ No se puede add a la BH

*/\**

*Recuerde. Un hecho es: var=valor*

*\*/*

**∴ R MT⟾**

Al tener el Mt muchos inconvenientes, los SE solo usan el MP.

Para subsanar esto, utilizan algoritmos del motor de inferencia, llamando Backward-Chaining (BWC).

# Algoritmos de encadenamiento (chaining algoritm’s)

//Algoritmos de redaccion en cadenas

Se los llama asi porque el disparo de una regla, puede hacer disparar otras reglas.

Ejemplo:

Billar Reglas

R1⟾h1

h2 R2⟾h3

R4⟾h4

**Existen 2 “criterios” para el encadenamiento.**

* **FWC (en profundidad)**
* **BWC (en anchura)**

## Forward Chainind (FWC)