Sistemas Expertos

Mauricio Elian Delgadillo Garcia

19 de agosto de 2020

Índice general

Evaluación Semestral

2 Parciales (Ambos = 80 %)

1 Examen Final
Proyecto (s) 20 %

Capítulo 1

Conceptos Básicos

1.1. Que es un Sistemas Expertos?

/* Sistema Experto (S. E.) = Sistema Especialista */

"Un S.E. es un software que emula el razonamiento humano actuando tal y cual lo haría un experto, en un área específica del conocimiento"

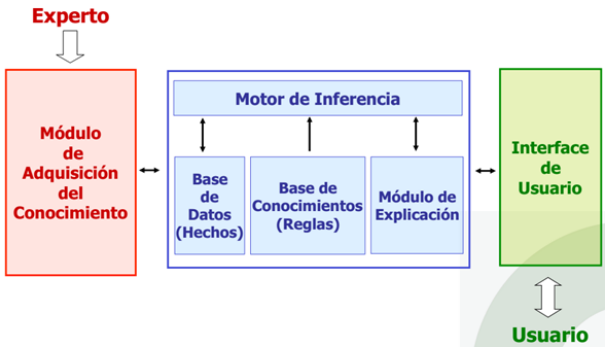
Un Experto Humano (E. H.), usualmente tiene un dominio muy amplio en el campo del saber, en el cual él se desempeña.

En constraste, un S. E. está dedicado a un área especifica de un campo del saber. Abarcar todo el conocimiento del E. H., sería muy complejo.

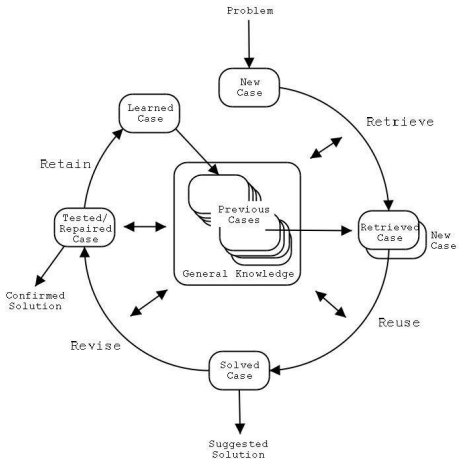
1.1.1. Tipos de Sistemas Expertos

En general, existen tres tipos de Sistemas Expertos (S. E.):

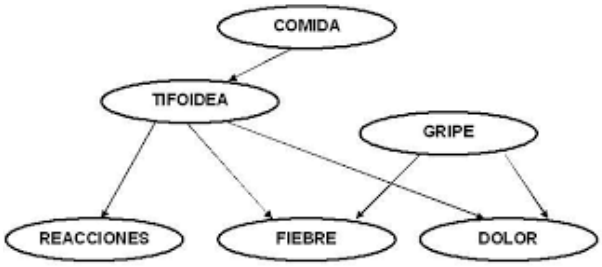
1. **S. E. Basados en Reglas o R. B. R. (Rule Based Reasoning)** .- Utilizan para el proceso de inferencia, un conjunto de Reglas de Producción que constituyen la Base de Conocimientos del S. E.



2. **S. E. Basados en casos o C. B. R. (Case Based Reasoning)** .- Proceso de solucionar nuevos problemas basándose en las soluciones de problemas anteriores.



3. **S. E. como Red Bayesiana** .- Modelo probabilístico que relaciona un conjunto de variables aleatorias mediante un grafo dirigido (Red).



1.1.2. Actores en el entorno de un S. E.

Sin tomar en cuenta el desarrollo del software, en el entorno de un Sistema Experto, intervienen tres actores:

- El Experto Humano (E. H.).- Comunica al I. C. el conocimiento que tiene sobre el área del S. E.
- El Ing. del Conocimiento (I. C.).- Estructura (Codifica) el conocimiento para que pueda ser almacenado en la Base de Conocimientos (B. C.) del S. E.
- El Cliente (Usuario Final).- Aquel que usa el Sistema para realizar consultas

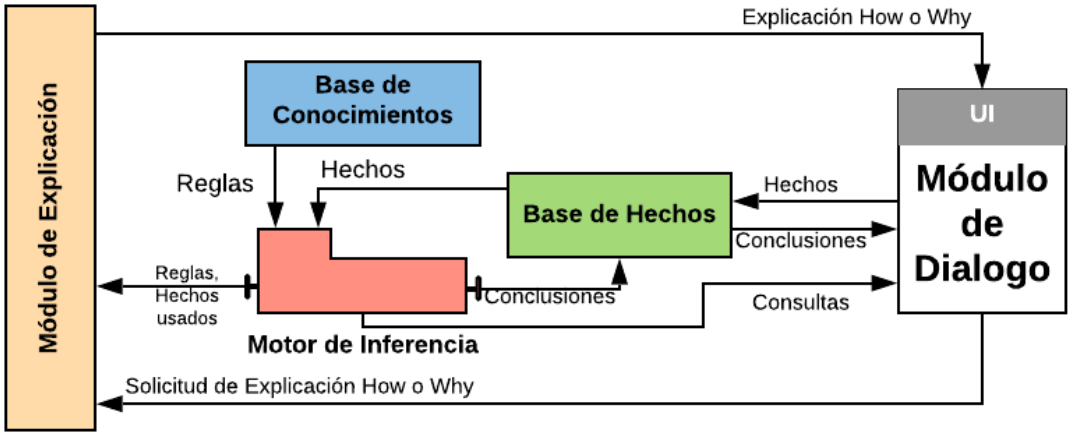
1.1.3. Ejemplos de S. E.

- El S.E. M.Y.C.Y.N. que determina las enfermedades de la sangre
- S.E. “Guía Turístico para Santa Cruz” Proyecto de Grado - U.A.G.R.M.
- S.E. “Para resolver problemas del motor Toyota Diesel” - Proyecto de Grado - U.C.B.

Otros Trabajos (Shell)

- Expert - GRM (UAGRM - 2005?) - Shell que usa compiladores y S.E.
- Shell UCB para S.E. Probabilísticos

1.2. Arquitectura de un S.E.



1.2.1. Modulo de Dialogo

Modulo de Diálogo = "Capa de Presentación".- Se le dice así, porque antiguamente era en modo texto. Es la UI (User Interface) del Sistema Experto. Permite:

1) Introducir **Hechos**, los cuales se almacenaran en la **Base de Hechos (B. H.)** y obtener de las B. H. las conclusiones (nuevos hechos) arrojadas por el **Motor de Interferencias (M. I.)**.

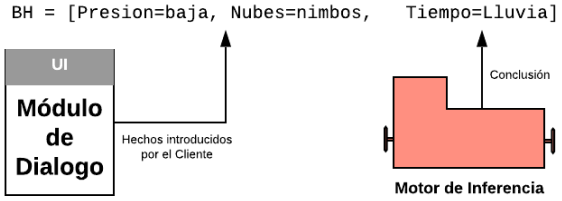
2) Solicitar y recibir del Módulo de Explicación, una explicación How o Why

3) Recibir una consulta (Pregunta) del **M. I.** El cliente puede omitir la respuesta o responder con un **Hecho**.

1.2.2. Base de Hechos

También llamada **Memoria de Trabajo**, la B. H., es un collection de Hechos, usualmente implementada como una Lista.

Un **Hecho** es una verdad asumida por el S. E. y proviene de dos fuentes: del **Cliente (final-user)** o del **Motor de Inferencias (M. I.)** como una conclusión. Textualmente, un hecho se representa como `Variable=valor`. Por ejemplo:



1.2.3. Motor de Inferencia (M. I.)

Contrasta los hechos de la **Base de Hechos (B. H.)** con el conocimiento almacenado en la **Base de Conocimiento (B. C.)**, intentando obtener conclusiones. Las conclusiones encontradas, son almacenadas en la **B. H.** como si fuesen "nuevos" hecho. El **M. I.** es básicamente un algoritmo que se basa en uno de éstos métodos:

- Forward-Chaining (F.W.C.)
- Backward-Chaining (B.W.C.)
- Hybrid Chaining (F.W.C. + B.W.C.)

1.2.4. Base de Conocimientos (B.C.)

El Ing. del Conocimiento divide y estructura el conocimiento del Experto Humano en ***piezas***, las cuales son almacenadas en la **B. C.**, asi, una **B. C.** es una **collection** persistente (i.e. que se guarda en la memoria secundaria) de éstas ***piezas***.

En particular, en un Sistema Experto Basado en Reglas o S.E.-R.B.R., cada pieza del conocimiento se almacena en una **Regla** de producción, la cual tienen la forma:

```
IF Premisa
 THEN
   Conclusion
```

1.2.5. Módulo de Explicación (M. E.)

El **M. E.** o **Explicador** almacena en Estructuras de Datos (E.D.), las piezas de conocimientos (Reglas en un R.B.R.) y los Hechos, que el **Motor de Inferencia (M.I.)** usa o ha usado durante la inferencia.

Así, cuando el **Cliente le solicita** una explicación, el **M.E.** pone a funcionar algoritmos que extraen de las E.D. la explicación pedida. Dos tipos de explicaciones pueden ser solicitadas al **M.E.**:

Explicaciones WHY

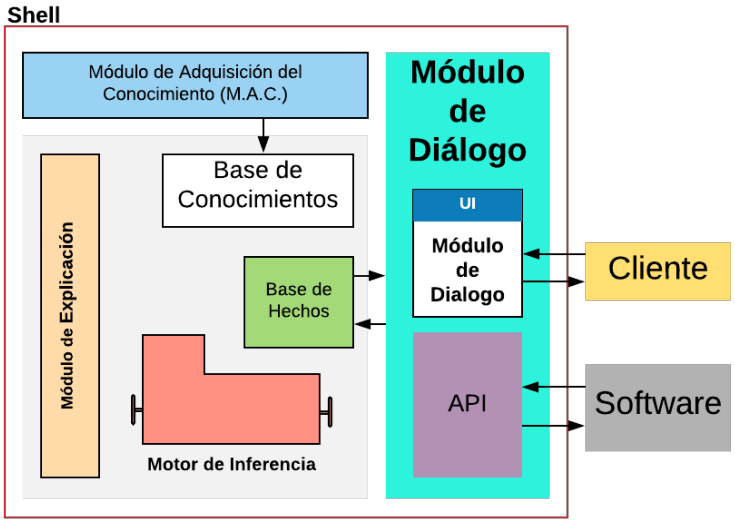
Esta aplicación puede ser lanzada por el cliente cuando el sistema experto le hace una pregunta, *S.E.,* ***Porque*** *me haces esa pregunta?* (La explicación WHY es la mas sencilla de implementar. Basta una Cola)

Explicaciones HOW

Esta aplicación solo es obtenida cuando el S.E. termina la inferencia (o consulta), *S.E.,* ***Como*** *llegaste a ésta conclusión?*

1.3. El Shell

Si una persona desarrolla un mismo tipo de S. E., pronto se dará cuenta del trabajo repetitivo que hará cada vez. Un **Shell** (caparazón, cubierta), para S. E., básicamente, implementa la Base de Hechos, el Motor de Inferencia, el Módulo de Explicación y la Base de Conocimiento (B. C.).



Como se observa, la Arquitectura de un Shell es similar a la de un S. E.

El Shell adiciona un ***Módulo de Adquisición del Conocimiento***, el cual le permite al Ing. del Conocimiento insertar el conocimiento, a la B. C.

También, el Shell crea un ***Módulo de Diálogo***, el cual puede tener una U. I. para un usuario humano y/o una A.P.I. para que un software interactúe con el S. E.

Capítulo 2

La Base de Conocimientos

Una Base de Conocimientos (B.C.) es como una "*base de datos*", que almacena piezas del conocimiento relacionadas y estructuradas, permitiendo la inserción y la lectura computarizada de las mismas. Esta Unidad se dedica a las B.C. de los S.E.-R.B.R. (**S.E. basados en Reglas**)

2.1. La B.C. en los S.E.-R.B.R.

En los S.E.-R.B.R. (Sistemas Expertos Basados en Reglas), las **piezas** del conocimiento se expresan como **Reglas de Producción**. Así, la Base de Conocimientos (**B.C.**) de un S.E.-R.B.R., es una **Collection de Reglas**.

Una **Regla**, textualmente, expresa una pieza del conocimiento como:

```
IF Premisa
 THEN
   Conclusion
```

Por ejemplo:

```
IF (Presion < 760) or (Temp > 80)
 THEN
   Hierve = Si
```

Nota.- Aunque una regla *textualmente* se exprese como un IF-THEN, en realidad NO se almacena así en la **B.C.** El diseñador de la B.C., usualmente, representara una regla como un objeto de una **Class Regla**

2.2. Variables

Antes de crear las Reglas, el Ing. del Conocimiento (I.C.) define *Variables*, con las cuales puede expresar *afirmaciones* sobre área específica al cual está dedicado el S.E.

Una *afirmación* se escribe como un factor-booleano:

```
[not] Variable OPREL Valor
```

Donde el **not** ($\neg$) es opcional y **OPREL** se refiere a los Operadores Relacionales: $<, >, =, \leq, \geq, \neq$. Por ejemplo, si el I.C. está haciendo un S.E. sobe Meteorología, podría definir las **variables**:

Presión - Presión atmósferica medida en mmg.

Nubes - Tipo de nube que se ve en el cielo

Con estas variables, el I.C. podrá hacer las afirmaciones:

```
Presion ≥ 760
¬(Nubes=cirros)
etc.
```

2.2.1. Tabla de Variables

Una vez definidas las **Variables**, el Ing. del Conocimiento (I.C.), debe determinar los valores que tomarán cada una de ellas.

Una herramienta **Formal** con la que dispone, es la `Tabla de Variables`, la cual expone en cada fila: El nombre de la Variable, su conjunto de valores y la restricción que se le aplica (opcional). Por ejemplo:

Variables	Conjuntos de Valores	Restricción
Talla	Chica, Media, Grande	
Edad	(numérico)	Edad ¿0
Peso	(numérico)	
Adulto	Si, No	

*(numérico): que manipula números Reales ($\mathbb{R}$)

Pero, **informalmente**, el I.C. puede especificar las valores de una variable, usando notación de conjuntos:

Talla $\in$ { Chica, Media, Grande }

Edad, Peso $\in \mathbb{R}$

Adulto $\in$ { Si, No }

2.2.2. Tipos de Variables

Observando la Tabla anterior, se identifican dos tipos de Variables:

Variables Numéricas

Son aquellas que usan como valor, cualquier número Real $\mathbb{R}$. Por ejemplo, las variables **Edad** y **Peso** son numéricas.

Las variables numéricas tienen la ventaja de poder usar **todos** los OPREL's ($<, >, =, \leq, \geq, \neq$), cuando forman un factor-boole. Por ejemplo: `Peso = 80, ¬Edad >100`

Variables Escalares

Son aquellas que tienen definido un **conjunto finito** de valores. Asi, las variables **Talla** y **Adulto** son escalares. Estas variables **solo** pueden utilizar los OPREL's = y $\neq$ (*igual* y *distinto*), cuando forman un factor-boole. Por ejemplo: `Talla = Chica, ¬Adulto ≠ No`

2.3. Con Lógica Proposicional

En la creación de las Reglas, el Ingeniero del Conocimiento (I.C.), utiliza propiedades de la Lógica Proposicional.

Para realizar un trabajo más cómodo con éstas propiedades, el I.C. convierte los factores boole en **literales** proposicionales.

2.3.1. Literal Proposicional

Un **literal** o **átomo**, es simplemente una letra proposicional, o su negación. Si la letra esta negada ($\neg$), se le dice literal negativo, caso contrario se le dice literal positivo. Ejemplos:

p //Literal positivo
¬q //Literal negativo
¬s //Literal negativo
t //Literal positivo

Note que el ¬ (not) es como el signo "-" en los números: Si está presente, es negativo; si no lo esta, es positivo

2.3.2. Factor Boole a Literales

Para renombrar un factor boole, simplemente se toma la parte `Variable=Valor` y se lo reemplaza con una letra. Note que en el reemplazo se respeta la negatividad del factor boole. Es decir, si el **factor boole es negativo**, también el **literal es negativo**

Por ejemplo, seria incorrecto reemplazar el factor boole negativo `¬(Edad>100)`, con el literal positivo **p**, Ejemplos de reemplazo:

¬(Edad>100)	por ¬**p**
Peso≠80	por **q**
Adulto=no	por **r**
¬Talla=chica	por ¬**s**

Note que no es necesario usar paréntesis cuando se aplica ¬ (not) a un factor boole.

2.3.3. Minterms

Se llama **minterm** o término mínimo a la conjunción (and,∧) de n ⩾ 1 literales. (*Entonces, un minterm, es simplemente una fila de literales conectados con* ∧)

Con literales, ejemplos de minterms:

¬s ∧ t ∧ ¬q	//de 3 Literales
t ∧ ¬p	//de 2 Literales
r	//de 1 Literal
¬q	//de 1 Literal

Observe que **un solo literal** es un **minterm**, porque en su definición dice: "*n ⩾ 1 literales*". Con factores boole, ejemplos de minterms: (*Basta usar factores boole, en vez de literales*)

x < 0 ∧ ¬M = si ∧ S > 0	//Min de 3 Literales
¬Peso < 0 ∧ Adulto = si	//Min de 2 Literales
¬Talla ≠ chica	//Min de 1 Literal

2.3.4. Forma Normal Disyuntiva (FND)

Se dice que una Fórmula proposicional está en su FND si y solo si, está expresada como una disyunción de n ⩾ 1 minterms. Recuerde que disyunción es **OR** lógico (∨). Así, un FND es una fila de minterms conectados ∨.

Naturalmente, en las FND de un minterm, será necesario encerrar entre paréntesis a aquellos minterm que tengan 2 o más literales.

Por ejemplo, con los 3 minterms: p ∨ q, z, s ∧ ¬t: se formaría la FND:

$$(p \wedge q) \vee z \vee (s \wedge \neg t)$$

Otros ejemplos con literales:

$(\neg s \wedge q \wedge r) \vee (\neg t \wedge r) \vee q \vee (r \wedge z)$	//FND de 4 minterms
$q \wedge \neg r \wedge p$	//FND de 1 Literal
p	//FND de 1 Literal

Obviamente, para construir FND's en el entorno de los S.E., usamos factores boole en vez de literales proposicionales. Algunos ejemplos usando Factores Boole:

- $(\neg Peso < 0 \wedge Adulto = si \wedge E = 0) \vee (x < 0 \wedge Polo = si) \vee (Talla = chica)$ //FND de 3 minterms
- $(Temp > 80 \wedge \neg Talla \neq grande \wedge x = 0 \wedge z < 0) \vee (x < 0 \wedge Presion = 0)$ //FND de 2 minterms
- $\neg Polo \neq no$ //FND de 1 minterm (un literal es un minterm)

Teorema 2.1. Para toda fórmula proposicional α, existe una formula proposicional β en FND, tal que $\alpha \equiv \beta$ (*Este teorema dice que **toda** fórmula puede ser "convertida" a su FND*)

Ejercicio: Llevar la fórmula

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

a su FND

Solución:

$$\begin{aligned} (p \vee q) \rightarrow (r \wedge \neg s) &\quad \equiv \text{Definición de Implicación} \\ \neg(p \vee q) \vee (r \wedge \neg s) &\quad \equiv \text{Ley de De Morgan} \\ (\neg p \wedge \neg q) \vee (r \wedge \neg s) & \end{aligned}$$

Se obtuvo una FND de 2 minterms.

Note que para convertir una fórmula a su Forma Normal Disyuntiva (FND), se necesita aplicar leyes del **Álgebra Proposicional**.

2.3.5. Maxterms

Se llama maxterm o término máximo a la disyunción (OR, $\vee$) de $\texttt{n} \geqslant \texttt{1}$ literales. (*Los maxterm se estudian teóricamente, pues en la práctica no son usados en la construcción de las Reglas.*)

Ejemplos de maxterms, usando literales:

$$\begin{array}{cc} t \vee \neg s \vee \neg q & \text{//de 3 literales} \\ r \vee \neg p & \text{//de 2 literales} \\ \neg r & \text{//de 1 literal} \\ s & \text{//de 1 literal} \end{array}$$

Advierta que un solo literal es simultáneamente: **maxterm, minterm** y **FND**

2.4. Reglas

En teoría, la **Premisa** y la**Conclusión** de una Regla pueden contener cualquier fórmula proposicional.

```
IF Premisa { Cualquier formula }
 THEN
   Conclusion { Cualquier formula }
```

Pero, para el Motor de Inferencia, le seria muy complicado, trabajar con éste tipo de reglas. Por este motivo se aplican restricciones.

2.4.1. Restricción a la Premisa

La Premisa de una Regla puede ser un **Minterm** o una FND. por ejemplo, usuando literales:

```
IF p ∧ q ∧ ¬s
 THEN
   Conclusion
```

```
IF (t ∧ q) ∨ (¬s ∧ ¬q) ∨ p
 THEN
   Conclusion
```

La primera regla usa como premisa un Minterm, mientras que la segunda, una FND

2.4.2. Restricción a la Conclusión

Las conclusión de una Regla debe ser atómica, es decir, solo puede ser **un literal** (*Recuerde que los literales, son los átomos proposicionales*)

Por ejemplo, las conclusiones de las siguientes reglas, son correctas:

```
IF Premisa
 THEN
    ¬p
```

```
IF Premisa
 THEN
    q
```

En contraste, por ejemplo, las siguientes reglas tienen conclusiones incorrectas:

```
IF Premisa
 THEN
    q ∨ ¬s
```

```
IF Premisa
 THEN
    (p ∧ ¬s) ∨ z
```

Al usar conectivos, la conclusión deja de ser atómica

Pero, cuando se trabaja con Reglas de un S.E., la conclusión de estas se restringe a:

La conclusión de una **Regla** de un S.E. sólo puede ser expresada como:
`Variable=Valor`

Por ejemplo, las siguientes Reglas de un S.E., tienen conclusiones correctas:

```
IF Premisa
 THEN
    Peso=50
```

```
IF Premisa
 THEN
    Adulto=si
```

A continuación, se ejemplifican Reglas de un S.E. con conclusiones incorrectas:

```
IF Premisa
 THEN
    ¬Peso=50
```

No se permiten literales negativos solo positivos

```
IF Premisa
 THEN
    Z ≠ no
```

Solo se admite el operador relaciona =

```
IF Premisa
 THEN
    M=si ∨ P=0
```

La conclusión no es atómica

2.5. Tipos de Reglas

Teniendo una Regla con una Premisa y una Premisa y una Conclusión correctamente formadas, el Tipo de Regla lo determina su Premisa:

- Si la Premisa es un **Minterm**, la Regla es simple o esencial.
- Si la Premisa es una **FND**, la Regla es compuesta.

El nombre que se le dan a los tipos de Reglas, en realidad depende del accionar del Motor del accionar del Motor de Inferencia (M.I.):

Cuando el M.I., desea obtener la conclusión de una **Regla esencial**, hace un algoritmo relativamente simple. Cuando el M.I. hace lo mismo con una **Regla compuesta**, hace **n** veces este algoritmo simple, donde **n** es la cantidad de Minterms que tiene la **FND** de la Premisa. Las Reglas Compuestas o Reglas-FND, son implementadas en los **Shells profesionales**

Ejemplos, usando letras proposicionales:

```
IF q∧¬s∧z
 THEN
    ¬m
```

Regla Simple. Su Premisa es un minterm.

```
IF (z∧¬s)∨z∨(p∧q∧t)
 THEN
    w
```

Regla Compuesta. Su Premisa es un FND (de 3 minterms)

```
IF s
 THEN
    ¬q
```

Regla Simple. Su Premisa es un minterm

Una Regla Simple es una Regla Compuesta? *Respuesta:* Claro que sí.

Dado que un minterm (la premisa de una regla simple), es también una FND, concluimos que una Regla Simple es también una Regla Compuesta. Naturalmente, una Regla Compuesta (con más de un Minterm) NO es una Regla Simple. Pero, afortunadamente, es posible reescribir una Regla Compuesta o cualquier otra, en una o más Reglas Simples, según lo declara el siguiente resultado:

Teorema 2.2. Para toda la regla R, existen $\mathtt{n} \geqslant \mathtt{1}$ Reglas Simples $R_1, R_2, ..., R_n$ tal que:

$$R \equiv \{R_1, R_2, ..., R_n\}$$

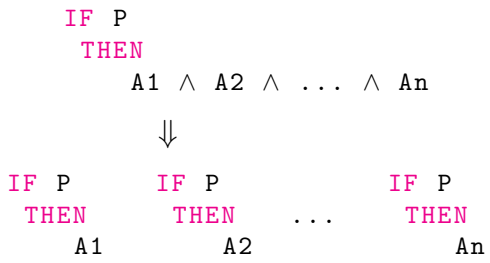
Informalmente, este Teorema afirma que ***toda*** *Regla R puede ser "convertida" en n Reglas Simples*

2.5.1. Conversión a Reglas Simples

Aunque aquí no probamos los Teoremas, tomaremos en cuenta la demostración del **Teorema 2.2.**, vista en la bibliografía afín. En esta demostración, se propone una Tabla con la cual se "'transforma" la premisa y la conclusión en las reglas simples:

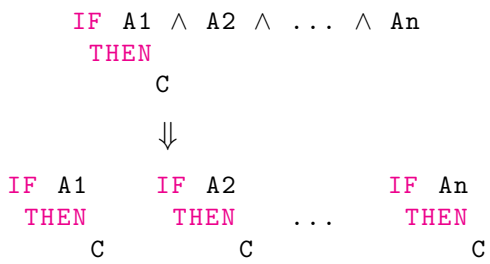
Tabla de Conversión

i) Si la Conclusión es una conjunción ($\wedge$) de $\mathtt{n}$ fórmulas.



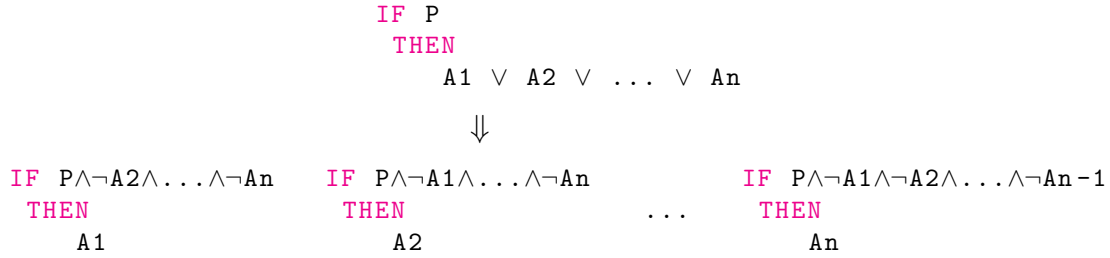
Cada fórmula de la conclusión forma su propia regla

ii) Si la Premisa es una disyunción ($\vee$) de `n` fórmulas.

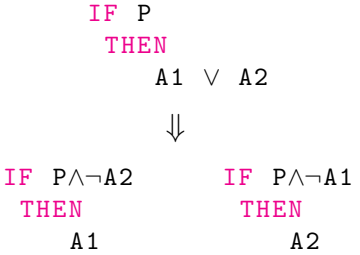


Cada fórmula de la premisa crea su propia regla, con la misma conclusión de la regla original.

iii) Si la Conclusión es una disyunción ($\vee$) de `n` fórmulas.



Es decir, una de las fórmulas se queda como conclusión, mientras que las otras "suben" negadas a la premisa, creando una conjunción. Por ejemplo, para dos fórmulas:



Ejercicio: Llevar R, a reglas simples.

```
(R) IF P
    THEN
      A1 ∨ A2 ∨ A3
```

Solución. Usando la Conversión *iii)* de la Tabla: R ≡ {R1,R2,R3}

```
(R1) IF P ∧ ¬A2 ∧ ¬A3
     THEN
       A1
```

```
(R2) IF P ∧ ¬A1 ∧ ¬A3
     THEN
       A2
```

```
(R3) IF P ∧ ¬A1 ∧ ¬A2
     THEN
       A3
```

Ejercicio: Convertir la siguiente regla R, en reglas simples:

```
(R) IF (p ∨ ¬q) → s
    THEN
      Z
```

Solución. La conclusión es atómica, no necesita arreglo. La premisa la llevamos a FND, antes de aplicar alguna conversación propuesta por la Tabla.

$$\begin{aligned} Premisa &\equiv (p \vee \neg q) \rightarrow s \\ &\equiv \neg(p \vee \neg q) \vee s \\ &\equiv (\neg p \wedge q) \vee s \end{aligned}$$