Universidad De Granada

Grado ingeniería informática (2016 – 2017)

---

# Ingeniería Conocimiento

## Práctica Final

Trabajo realizado por: Antonio Miguel Morillo Chica con DNI 77379021Y

---

# 1. Introducción.

La creación de sistemas de conocimiento que se enfocan a un tema determinado un punto muy interesante. Son sistemas que actúan e intentan parecerse en un tema. Estos, si se centran en un ambito se les denomina sistema experto.

En nuestro caso nuestro SE es un sistema dedicado al tratamiento de licencias y de datos. El sistema pose tres módulos diferenciados:

- Para recomendar licencia: Tras una serie de preguntas es capaz de detectar la licencia que más conviene según las características elegidas. Para saber las preguntas habremos tenido que crear un árbol de decisión, lo explicaremos más adelante.
- Para ver incompatibilidades: Introduciremos dos licencias y nos dirán si son compatibles o no por alguna razón. En este módulo el sistema deberá detectar que es o no compatible a través de una batería de ejemplos, es por tanto un sistema supervisado basado en reglas.
- Para tratar información sensible y categorizarla.

Para moverse entre módulos usamos un menú en el que cada opción representa la incursión en una de estas "zonas". Este mismo menú fue desarrollado para los ejercicios de entrega.

# 2. Proceso del desarrollo del conocimiento.

El proceso seguido lo podemos dividir en dos etaapas, por un lado la adquisición del conocimiento y por otro lado la validación del mismo en el sistema.

## 2.1. Adquisición del conocimiento.

La adquisición del conocimiento la podemos hacer mediante dos fuentes, a través de un experto y por otro del análisis de textos. En mi caso la gram mayoría del conocimiento ha sido obtenido del análisis de textos.

Los documentos fundamentales han sido los obtenidos en la plataforma DECSAI como base. Por otro lado la resolución de dudas y matizaciones de los conceptos y características obtenidas han sido comparadas y cambiadas según algunas páginas.

Para la adquisición del conocimiento, una vez aprendidos los conceptos claves y entender el tema a tratar he usado la técnica de la rejilla o emparrillado, técnica usada en psicología para el análisis del proceso mental de los pacientes.

### 2.1.1. Conocimiento del modulo uno.

Para el conocimiento del modulo, como he mencionado he usado la tecinica de la rejilla. En primer lugaar he seleccionado 5 licencias (aunque sea el mínimo esto es debido a la gran ambigüedad en el tratamiento de licencias, a la gran variedad de la misma y a la dificultad de la valoración de sus caracteristicas), estas son:

- GPL: GNU General Public Licence.
- LGPL: Lesser General Public Licence.
- MIT/X11: Massachusetts Institute of Technology
- MPL: Mozilla Public License.
- RP: Reciprocal Public.

La obtención de sus características las podemos obtener del documento aportado y de páginas como choosealicense.com y oss-watch.ac.uk/apps/licdiff/. Yo he optado por las siguientes:

- Libre: proporciona la libertad de ejecutar el programa, para cualquier propósito; estudiar el funcionamiento del programa, y adaptarlo a sus necesidades, redistribuir copias; mejorar el programa, y poner sus mejoras a disposición del público, para beneficio de toda la comunidad.
- Abierta/OpenSource: Distribución libre; inclusión del código fuente; permitir modificaciones y trabajos derivados en las mismas condiciones

que el software original; integridad del código fuente del autor, pudiendo requerir que los trabajos derivados tengan distinto nombre o versión, etc.

- CopyLeft: términos de distribución no permiten a los redistribuidores agregar ninguna restricción adicional cuando lo redistribuyen o modifican, o sea, la versión modificada debe ser también libre
- Patente: conjunto de derechos exclusivos garantizados por un gobierno o autoridad al inventor de un nuevo producto (material o inmaterial) susceptible de ser explotado industrialmente para el bien del solicitante por un periodo de tiempo limitado.

Tras aplicar el emparrillado la tabla de elementos/características queda de la siguiente forma:

| | Libre | Open Source | CopyLeft | Patente |
|---|---|---|---|---|
| GPL | 3 | 3 | 3 | 3 |
| LGPL | 3 | 2 | 2 | 3 |
| X11 | 1 | 2 | 0 | 0 |
| MPL | 3 | 2 | 1 | 3 |
| Reciprocal Public | 0 | 1 | 3 | 3 |

Como segundo paso de la regilla compruebo la distancia entre los elementos y selecciono la menor de todas.

| | GPL | LGPL | X11 | MPL | Reciprocal Public |
|---|---|---|---|---|---|
| GPL | | 2 | 9 | 3 | 5 |
| LGPL | | | 7 | 1 | 5 |
| X11 | | | | 6 | 8 |
| MPL | | | | | 6 |
| Reciprocal Public | | | | | |

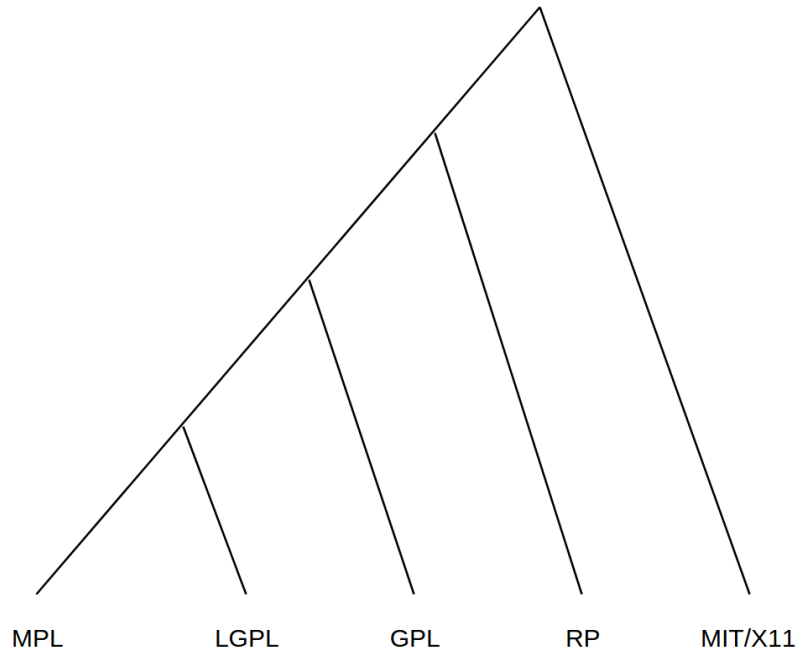
Tras seleccionar la menos distancias juntaaremos MPL con LGPL, para la distancia entre GLP, X11 y RP se seleccionará aquella entre MPL y LGPL que tengan menor distancia. Este proceso lo aplicaremos varias veces hasta que solo queden dos “elementos”:

| | GPL | X11 | Reciprocal Public | LGPL-MPL |
|---|---|---|---|---|
| GPL | | 9 | 5 | 2 |
| X11 | | | 8 | 6 |
| Reciprocal Public | | | | 5 |
| LGPL-MPL | | | | |

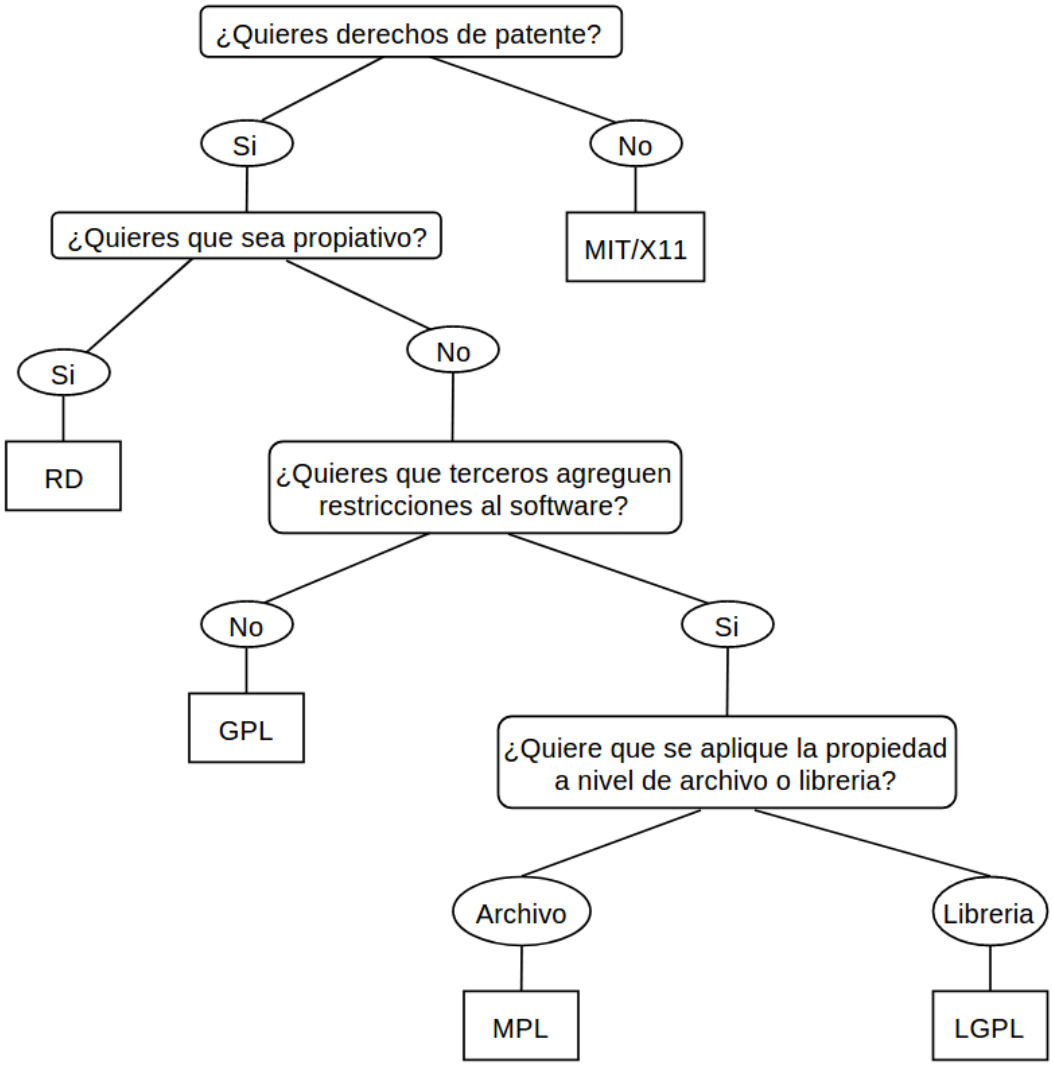
| | X11 | Reciprocal Public | LGPL-MPL-GPL |
|---|---|---|---|
| X11 | | 8 | 6 |
| Reciprocal Public | | | 5 |
| LGPL-MPL-GPL | | | |

| | X11 | LGPL-MPL-GPL-RP |
|---|---|---|
| X11 | | 6 |
| LGPL-MPL-GPL-RP | | |

Tras esto nos quedaría árbol de las patentes en el que cada rama será consecuencia de una característica diferenciadora:




Finalmente el árbol de decisión que he realizado en el módulo 1 es el siguiente, las hojas son nodos solución:




Este es árbol de decisiones que uso para implementar el módulo 1. Que muestra como desplazarse entre un árbol de decisiones.

### 2.1.2. Conocimiento del modulo dos.

Para la adquisición del conocimiento del módulo dos lo primero que he hecho ha sido intentar abstraer la razón de la incompatibilidad de las licencias. Primero he montado una serie de 15 ejemplos entres las diferentes licencias. Por ejemplo:

- GPL se le añade MIT/X11
- GPL se le añade LGPL

- MIT/X11 se le añade RD
- MIT/X11 se le añade MPL
- étc.

En la siguiente imagen podemos observar algunos de los ejemplos usados, no todos.

| | MPL | GPL | Compatibilidad | |
|---|---|---|---|---|
| Libre | 3 | 3 | 1 | 0 |
| OS | 2 | 3 | 1 | |
| CopyLeft | 1 | 3 | 1 | |
| Patente | 3 | 3 | 0 | |

| | MPL | X11 | Compatibilidad | |
|---|---|---|---|---|
| Libre | 3 | 1 | 1 | 1 |
| OS | 2 | 2 | 1 | |
| CopyLeft | 1 | 0 | 1 | |
| Patente | 3 | 0 | 1 | |

| | GPL | RD | Compatibilidad | |
|---|---|---|---|---|
| Libre | 3 | 0 | 0 | 0 |
| OS | 3 | 1 | 0 | |
| CopyLeft | 3 | 3 | 0 | |
| Patente | 3 | 3 | 0 | |

| | X11 | RD | Compatibilidad | |
|---|---|---|---|---|
| Libre | 1 | 0 | 1 | 0 |
| OS | 2 | 1 | 0 | |
| CopyLeft | 0 | 3 | 1 | |
| Patente | 0 | 3 | 0 | |

| | LGPL | X11 | Compatibilidad | |
|---|---|---|---|---|
| Libre | 3 | 1 | 1 | 1 |
| OS | 2 | 2 | 1 | |
| CopyLeft | 2 | 0 | 1 | |
| Patente | 3 | 0 | 1 | |

| | MPL | RD | Compatibilidad | |
|---|---|---|---|---|
| Libre | 3 | 0 | 0 | 0 |
| OS | 2 | 1 | 0 | |
| CopyLeft | 1 | 3 | 1 | |
| Patente | 3 | 3 | 0 | |

Los valores de cada licencia es una estimación de a mi juicio de los valores que representa cada característica donde:

- Libertad: Grado de poder distribuir, uso y aprendizaje. 1 no se especifica o es permisivo, 0 no es libre, 2 es libre pero hay limitaciones de uso o distribución, 3 para cuando se considera libre totalmente.
- OS: OpenSource, que el codigo sea abierto y que se permita modificaciones. 1 no se especifica, 0 no es abierto, 2 es abierto y 3 es para cuando es el coyleft weak y strong. Esto último es debido a que se pretende fomentar la idea de soft. Libre.
- CopyLeft: Grado en el que se el copyleft, 0 para permisivo, 1 para weak parcial, 2 weak compuesto y 3 strong.

La columna cuatro representa si la característica es compatible según el rango del valor, así, una licencia con copyleft débil no puede aceptar una licencia con copyleft fuerte. La última columna representa pues si la licencia en general es compatible o no, esta información ha sido optenida de http://operaciones.cenatic.es/comunidad-legal/CalculadorLicencias. Una herramienta del gobierno de España, concretamente de la Comunidad de Aspectos Legales del Software Libre.

Finalmente he reunido los resultado en una tabla más específica donde observo el grado diferencia o distancia entre el valor de una característica y el número de incompatibilidades que genera. Si la distancia es muy grande quiere decir que ha provocado un sesgo importante y que son las primeras características a comparar y a consultar para descalificar la compatibilidad o incompatibilidad.

Aquí adjunto las tablas realizadas para que se vea más claramente.

| | Valor | Compatible | Incompatible |
|---|---|---|---|
| Libre | 0 | 1 | 3 |
| | 1 | 1 | 2 |
| | 2 | 4 | 2 |
| | 3 | 5 | 2 |
| Open Source | 0 | | |
| | 1 | 3 | 0 |
| | 2 | 5 | 3 |
| | 3 | 2 | 1 |
| CopyLeft | 0 | 3 | 0 |
| | 1 | 3 | 0 |
| | 2 | 3 | 1 |
| | 3 | 1 | 5 |
| Patente | 0 | 0 | 3 |
| | 1 | | |
| | 2 | | |
| | 3 | 4 | 8 |

| Tcompatibles | Tincomatibles | Distancia |
|---|---|---|
| 12 | 9 | 3 |
| 10 | 5 | 5 |
| 11 | 5 | 7 |
| 4 | 11 | 7 |

# 3. Descripción del sistema.

El sistema consta de tres módulos diferenciados.

- **Módulo 1**: Módulo encargado de recomendar una licencia usando un árbol binario de preguntas respuestas. La adquisición del conocimiento del árbol ha sido realizado por la técnica de la rejilla vista en clase de teoría.

Los datos de entrada y salida y el uso de las reglas las definiremos en el siguiente punto.

- **Módulo 2**: Módulo encargado de ver si dos licencias son compatibles. El árbol de decisión para la comparación de características la he realizado usando la distancia inherente a la compatibilidad individual de cada variable en el con su semejante de otra característica. Esta valoración ha sido abstraída de articulos de GNU, wikipedia y herramientas de organismos reconocidos como el mencionado anteriormente.
- **Módulo 3**: Módulo no desarrollado por incomprensión y falta de tiempo para su desarrollo.

## 3.1. Variables de entrada/salida.

Las variables de entrada y salida las definiremos en una tabla para que toda la información esté más compacta y sea más fácil leerla:

| | Entrada | Salida |
|---|---|---|
| Modulo 1 | Respuestas binarias a las preguntas respondidas por el sistema: si o no. | • Pregunta realizada por el sistema que representa un nodo de decisión.<br>• Licencias restantes de las que sumimos cuales deben seguir siendo valoradas.<br>• Licencia final compatible y explicación de porque lo es. |
| Modulo 2 | • Variable 1: Licencia que se tiene en el software.<br>• Variable 2: Licencia que se quiere añadir al software. | Explicación de porque la licencia es o no compatible con la aportada en primer lugar. |

## 3.2. Conocimiento inicial del sistema.

El sistema poseía inicialmente para el módulo 1 la lista de licencias para mostrar en cada paso del árbol de decisión la lista de licencias que quedaban para ser analizadas. Esta característica finalmente ha sido quitada por lo que no se puede considerar conocimiento inicia.

Por otro lado se tiene la lista de licencias con las características nombradas anteriormente, libre, open source, copyleft y patente. Para representar esto he usado el template Licencia:

```
(deftemplate Licencia
  (field nombre)
  (field libre)
  (field os)
  (field copyleft)
  (field patente)
)
```

Evidentemente para introducir estos hechos iniciales hay que establecer los facts siguientes, sí añadimos más licencias por un lado habría que introducirlos también aquí además de reestructurar el árbol de decisiones.

```
(deffacts listaCaracteristicas
  (Licencia (nombre GPL)     (libre 3)  (os 3)  (copyleft 3) (patente 3))
  (Licencia (nombre LGPL)    (libre 3)  (os 2)  (copyleft 2) (patente 3))
  (Licencia (nombre MPL)     (libre 1)  (os 2)  (copyleft 0) (patente 0))
  (Licencia (nombre MIT/X11) (libre 3)  (os 2)  (copyleft 1) (patente 3))
  (Licencia (nombre RP)      (libre 0)  (os 1)  (copyleft 3) (patente 3))
)
```

## 3.3. Estructura del razonamiento del sistema.

El razonamiento está muy delimitado debido a que los módulos desarrollados se desarrollan casi prácticamente secuencial debido a la estructura de árbol en la toma de decisiones.

Los módulos únicamente se ejecutarán en el momento en el que el usuario seleccione en el menú que desea modificar, una vez hecho esto inevitablemente se

dispararán reglas que tengan que ver con este módulo y sin excepción reglas de otros módulos.

## 3.4. Reglas del sistema.

Existen tres reglas iniciales que se ejecutarán siempre al arrancar el sistema experto que son:

```
(defrule inicio
  =>
  (printout t "Este es el sistema experto para licencias y proteccion de datos." crlf )
  (printout t "Que desea hacer?" crlf )
  (assert (inicio-realizado))
)

(defrule elegirOpcion
  ?m <- (inicio-realizado)
  =>
  (printout t "    1: Elegir licencia" crlf )
  (printout t "    2: Comparar licencia" crlf )
  (printout t "    3: Protección de datos" crlf)
  (printout t crlf)
  (printout t " -> ")
  (bind ?opcion (read))
  (assert (opcion-elegida ?opcion))
  (retract ?m)
)

(defrule mostrarOpcion
  (opcion-elegida ?eleccion)
  =>
  (if (and (> ?eleccion 0) (< ?eleccion 4)) then
    (if (= ?eleccion 1) then (assert (modulo-uno)))
    (if (= ?eleccion 2) then (assert (modulo-dos)))
    (if (= ?eleccion 2) then (assert (modulo-tres)))
  else
    (printout t "Opcion incorrecta." crlf)
    (assert (inicio-realizado))
  )
)
```

Inicio se dispara automáticamente nada más arrancar ya que se dispara sin ningún hecho existente. Posteriormente se muestra el menú para saber el módulo a ejecutar. La tercera regla decide en función del módulo seleccionado la regla introductoria que disparará el encadenamiento de reglas de cada módulo.

Para el módulo he usado las siguientes reglas, hay que tener en cuenta que se ejecutan secuencialmente:

- Pregunta 1: Elimina las licencias que no tienen derecho a patente. El

antecedente se da cuando se selecciona en el menú el modulo 1.

```
(defrule questionOne
  (modulo-uno)
  =>
  (printout t crlf)
  (printout t "MODULO 1" crlf)
  (printout t "En este modulo induciremos cual es la licencia que más se adapta a sus necesidades,
según una serie de preguntas clave. No serán muchas :3" crlf)
  (printout t crlf)

  (printout t "¿Quiere derecho de patente? si/no" crlf)
  (bind ?respuesta (read))
  (if (eq ?respuesta si) then (printout t "Restantes: Republic Domain, MPL, LPGL y GLP")  (printout t
crlf))
  (assert (respuesta-uno ?respuesta))
)
```

- Pregunta 2: Elimina todas las licencias que no sea te corte propietario o si lo sea. El antecedente se activa siempre y cuando se haya ejecutado la regla 1.

```
(defrule questionTwo
  (respuesta-uno ?r)
  =>
  (printout t crlf)
  (printout t "¿Quieres que sea propietario? (si/no)" crlf)
  (bind ?respuesta (read))
  (if (eq ?respuesta no) then (printout t "Restantes: MPL, LPGL y GLP") (printout t crlf))
  (assert (respuesta-dos ?respuesta))
)
```

- Pregunta 3: Elimina las licencias según lo duro que sean cambiar sus restricciones:

```
(defrule questionThree
  (respuesta-dos ?r)
  =>
  (printout t crlf)
  (printout t "¿Quieres que se se use tu software y terceros añadan restricciones? (si/no)" crlf)
  (bind ?respuesta (read))
```

```
  (if (eq ?respuesta si) then (printout t "Restantes: MPL y LPGL")  (printout t crlf))
  (assert (respuesta-tres ?respuesta))
)
```

- Pregunta 4: Decide finalmente la licencia se aplica a nivel de módulo o a nivel de librería.

```
 (defrule question-for
  (respuesta-tres ?r)
  =>
  (printout t crlf)
  (printout t "¿Quiere que se aplique la propiedad a nivel de archivo o libreria?" crlf)
  (bind ?respuesta (read))
  (assert (respuesta-cuatro ?respuesta))
)
```

- Resuolución del arbol: En cada pregunta hemos guardado la respuesta. Por último se ejecutará la regla que resuelve y aplica el conocimiento adquirido de todas las respuestas decidiendo que mostrar y el porqué de la licencia.

```
(defrule arbol-decision
  (respuesta-uno ?r1)
  (respuesta-dos ?r2)
  (respuesta-tres ?r3)
  (respuesta-cuatro ?r4)
  =>
  (printout t crlf)
  (if (eq ?r1 no) then
    (if (eq ?r2 si) then
      (printout t "La mejor licencia es: Republic Domain" crlf)
      (printout t "Razón: Es una licencia permisiva por lo que puede ser libre o privada." crlf)
    else
      (if (eq ?r3 si) then
        (if (eq ?r4 archivo) then
          (printout t "La mejor licencia es: MPL" crlf)
          (printout t "Razón: Es una licencia de medium copyleft a nivel de archivos" crlf)
        else
          (printout t "La mejor licencia es: LPGL" crlf)
          (printout t "Razón: Es una licencia de low copyleft a nivel de librerias" crlf)
        )
      else
        (printout t "La mejor licencia es: GLP" crlf)
        (printout t "Razón: La licencia GLP es muy restrictiva con el copyleft por lo que si tu
proyecto no cambirá sus restricciones si lo usan terceros." crlf)
      )
    )
  else
```

```
    (printout t "La mejor licencia es: MIT/X11" crlf)
    (printout t "Razón: Es la única licencia sin derechos de patente." crlf)
  )
)
```

Las reglas para el módulo dos siguen esta misma estructura secuencial ya que el árbol de decisión lo he obtenido de la forma descrita en el punto 2. Primero leo las licencias que quiero analizar para lo que uso dos reglas:

```
(defrule licenciaOne
  (modulo-dos)
  =>
  (printout t crlf)
  (printout t "MODULO 2" crlf)
  (printout t "En este modulo induciremos la impativilidad de dos licencias" crlf)
  (printout t crlf)

  (printout t "LicenciaTipo I: " )
  (bind ?respuesta (read))
  (assert (respuesta-m21 ?respuesta))
)

(defrule licenciaTwo
  (respuesta-m21 ?l1)
  =>
  (printout t "LicenciaTipo II: " )
  (bind ?respuesta (read))
  (assert (respuesta-m22 ?respuesta))
  (assert (comparar-copyleft))
)
```

Por último la idea seguida para elegir la licencia en función de las características es comprar el valor con el que han sido intrducidas:

```
(defrule copyleft
  (comparar-copyleft)
  (respuesta-m21 ?l1)
  (respuesta-m22 ?l2)
  (Licencia (nombre ?l1) (copyleft ?cl1))
  (Licencia (nombre ?l2) (copyleft ?cl2))
  =>
  (if (<= ?cl1 ?cl2)
    then
      (assert (comparar-patente))
    else
      (printout t "Licencias incompatibles porque el copyleft de " ?l1 " es muy restrictivo" crlf)
  )
)

(defrule patente
  (comparar-patente)
  (respuesta-m21 ?l1)
  (respuesta-m22 ?l2)
  (Licencia (nombre ?l1) (patente ?p1))
```

```
  (Licencia (nombre ?l2) (patente ?p2))
  =>
  (if (and (= ?p1 0) (> ?p2 1))
    then
      (printout t "La licencia " ?l2 " no es compatible con la " ?l1 " porque " ?l1 " no tiene
derechos de patente a diferencia de " ?l2 crlf)
    else
      (if (and (= ?p2 0) (> ?p1 1))
        then
          (printout t "La licencia " ?l1 " no es compatible con la " ?l2 " porque " ?l2 " no tiene
derechos de patente a diferencia de " ?l1  crlf)
        else
          (printout t "La licencia " ?l2 " es compatible a nivel de patente " ?l1 crlf)
          (assert (comparar-os))
      )
  )
)

(defrule so
  (comparar-os)
  (respuesta-m21 ?l1)
  (respuesta-m22 ?l2)
  (Licencia (nombre ?l1) (os ?os1))
  (Licencia (nombre ?l2) (os ?os2))
  =>
  (if (and (= ?os1 1) (> ?os2 1))
    then
      (printout t "La licencia " ?l2 " no es compatible con " ?l1 )
      (printout t " debido a que " ?l1 " no es una licencia para soft propietario y " ?l2 " es para
soft libre " crlf)
    else
      (if (and (> ?os1 1) (= ?os2 0))
        then
        (printout t "La licencia " ?l2 " no es compatible con " ?l1 )
        (printout t " debido a que " ?l1 " es para soft libre y " ?l2 " es una licencia para soft
propietario  " crlf)
        else
          (assert (comparar-libertad))
      )

  )
)

(defrule libre
  (comparar-libertad)
  (respuesta-m21 ?l1)
  (respuesta-m22 ?l2)
  (Licencia (nombre ?l1) (libre ?f1))
  (Licencia (nombre ?l2) (libre ?f2))
  =>
  (if (= ?f2 0)
    then
      (printout t "La licencia " ?l2 " es privada no son compatibles" crlf)
    else
      (if (> ?f1 ?f2)
        then
          (printout t "Compatibles a nivel de libertad")
        else
          (printout t "No son compatibles a nivel de libertad" crlf)
      )
  )
)
```