FACULTAD DE INFORMÁTICA

LICENCIATURA EN DOCUMENTACIÓN

SISTEMAS DE REPRESENTACIÓN

Y

PROCESAMIENTO AUTOMÁTICO DEL CONOCIMIENTO

PRÁCTICA 3

Sistemas Basados en el Conocimiento: Gestión de una Librería Especializada en Cómics

Abril 2003

1. Objetivo

La presente práctica tiene como objetivo el desarrollo de un pequeño sistema basado en el conocimiento por medio de la herramienta PROTEGE junto con el entorno de sistemas de producción JESS (el cual es una implementación de CLIPS en JAVA), llevando a la práctica los conocimientos vistos en las clases teóricas de la asignatura.

2. Desarrollo de la práctica.

Para el desarrollo de un sistema basado en el conocimiento deberemos disponer de una base de conocimiento formada en este caso por una estructura de "frames", además deberemos de disponer de un conjunto de reglas que modelen la forma de poder añadir nuevo conocimiento en el sistema de forma dinámica.

3. Descripción del problema: Librería Especializada en Cómics

El siguiente problema trata de representar la gestión de la información de una tienda de cómics. Los cómics se han convertido hoy en día en artículos de culto por parte de muchas personas, lo cual ha generado que alrededor de un título aparezcan numerosos productos de *merchandising* como pósters, pins, camisetas y demás. Todo ello complica más la gestión de este tipo de tiendas.

Entrando en detalle en el problema que nos ocupa, una librería especializada en cómics ofrece los siguientes productos:

- Comic-Books: de los cuales habrá que guardar obligatoriamente información del título, número del mismo y de la cantidad de ejemplares existentes, así como quienes son el guionista y el dibujante de dicho número.
- Trading Cards: más comúnmente conocidos como "cromos".
- Pósters: hay que guardar el título, las medidas (ancho x largo) y el dibujante.
- Camisetas: hay que guardar la talla, distinguir entre manga corta y manga larga.
- Revistas: título, periodicidad.
- Figuras: título, medidas (ancho x largo x alto).
- Mugs: título.
- Pins: título.
- Cintas de vídeo: título, formato (PAL, ...), duración (en minutos).
- Otros (zippos, bolígrafos, maquetas, caretas,....): de los cuales guardaremos el título.

Para cada producto habrá que almacenar una referencia que lo distinga del resto, y claro está, el precio. Además, habrá que indicar los productos que hay actualmente en stock.

Los productos que se suelen vender en una librería especializada en cómics se dividen en tres **zonas** claramente diferenciadas de acuerdo a la procedencia de dicho producto. Estas zonas son:

- zona americana.
- zona europea.
- zona manga.

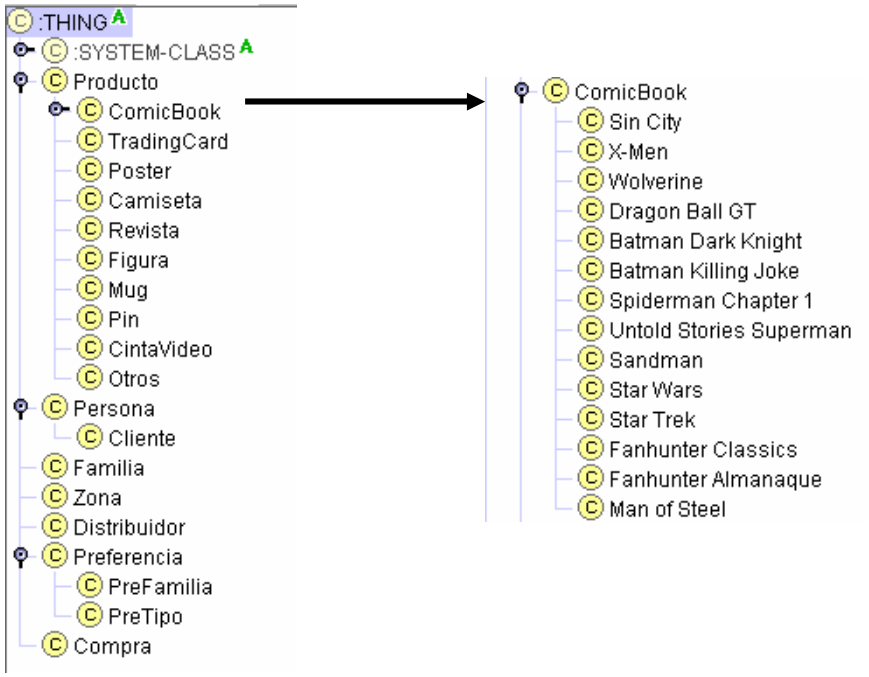
Será necesario conocer para cada producto a qué zona pertenece.

Además de la indicación referente a la zona de un producto, éstos deben pertenecer a una **familia** determinada. Una familia nos indicará una determinada línea de productos. Por ejemplo, podríamos hablar de la familia de productos sobre "Batman" o de la familia de productos sobre "Tintín". Para cada familia será necesario guardar información del nombre de la familia, su descripción y la zona a la que pertenece.

Aparte de esta información de los productos, en la tienda también se debe controlar la información referente a los **distribuidores** de los diferentes productos. De cada distribuidor se desea disponer de su nombre, dirección, identificación del distribuidor, así como una indicación de las posibles zonas que distribuye.

Por último, será necesario guardar información referente a los **clientes** habituales de la tienda, para los cuales se desea conocer sus gustos (en cuanto a familias de productos) y una lista de tipos de productos en los que está interesado. También se almacenará para cada cliente su nombre, apellidos, dirección y un identificador de cliente.

En la siguiente figura se muestra la estructura de frames propuesta en Protege:



Dentro de esa estructura es necesario comentar dos clases:

- Compra: en esta clase se almacenará la última compra realizada en la tienda. Dicha información será borrada una vez ha sido analizada. Sus slots son:

- Cli: persona que acaba de comprar un producto.
- Prod: instancia del producto comprado.
- Unidades: número de unidades compradas.
- ValidadoProducto: booleano que inicialmente está a FALSE y que se pone a TRUE cuando se han analizado las preferencias respecto al producto comprado (más detalles al final de la práctica).
- ValidadoFamilia: booleano que inicialmente está a FALSE y que se pone a TRUE cuando se han analizado las preferencias respecto a la familia del producto comprado (más detalles al final de la práctica).

- Clase Preferencia: consiste en una clase donde almacenar las distintas compras de productos por los clientes organizados por Familias o por Tipo de producto (Subclases: PreFamilia y PreTipo respectivamente). Es como un histórico de preferencias por parte de los clientes de la tienda. Esta información será empleada para conocer los gustos de los clientes. Los slots son los siguientes:
  - Comprador: cliente que ha comprado algo.
  - Gusta: producto o familia de la cual el cliente alguna vez ha comprado algo.
  - Cantidad: sumatorio de todas las compras realizadas por el cliente del producto o de la familia correspondiente.
  - Definitivo: booleano que estará a FALSE inicialmente y deberá pasar a TRUE cuando se considere que al cliente realmente le gusta ese producto o familia.

3.1. Datos del problema

En este punto se indica la información de la que se dispone al inicio del sistema, evidentemente dicha información deberá irse modificando a medida que en la tienda especializada se vayan produciendo eventos como venta de productos, adquisición, etc.

NOTA: Todos los productos tienen un campo fecha (en el que llegó el producto a la tienda) y un campo con el stock actual cuyos valores pueden ser inventados por el alumno.

- **Distribuidores**

Nombre	Dirección	Identificador	Zonas
Norma	Barcelona	Norma	americana, europea, manga
DistrImagen	Madrid	DistrI	americana, europea, manga
Diamond	USA	Diamond	americana
Another Universe	USA	Another	americana

- **Clientes**

Nombre	Dirección	Id.	Familias	Productos
Carlos ThreeC	C. Guanajuato	3	Batman, Spiderman, Sin City, X-Men, Sandman	Comic-Books, Mugs, Pins
José King	C. Sésamo	1	Sin City, X-Men, Sandman, Star Wars	Comic-Books, Pósters, Camisetas, Cintas de Vídeo.

- **Familias**

Nombre	Zona	Descripción
Batman	americana	Productos relacionados con el personaje creado por Bob Kane.
Superman	americana	Productos relacionados con el personaje principal de la editorial DC.
Spiderman	americana	Productos relacionados con el trepamuros creado por Stan Lee y Steve Ditko.
Sin City	americana	Productos relacionados con la ciudad creada por Frank Miller.
X-Men	americana	Productos relacionados con los mutantes odiados y temidos por la sociedad, creados por Stan Lee y Jack Kirby.
Dragon Ball	manga	Productos relacionados con el manga de Katsuhiro Otomo.
Fanhunter	europea	Productos relacionados con la obra creada por Cels Piñol.
Sandman	americana	Productos relacionados con el personaje de Morfeo y su entorno, creado por Neil Gaiman.
Star Wars	americana	Productos relacionados con la saga de ciencia ficción creada por George Lucas.
Star Trek	americana	Productos relacionados con la serie de ciencia ficción ideada por Gene Roddenberry.

- **Comic-Books**

Título	Nº	Cantidad	Familia	Guionista	Dibujante
Sin City	1, 2		Sin City	Frank Miller	Frank Miller
X-Men	1, 2, 3, 4		X-Men	Chris Claremont	Jim Lee
Wolverine	1, 2, 3		X-Men	Chris Claremont	John Buscema
Dragon Ball GT	20		Dragon Ball	Katsuhiro Otomo	Katsuhiro Otomo
Batman: The Dark Knight Returns	4		Batman	Frank Miller	Klaus Janson
Batman: The Killing Joke	1		Batman	Alan Moore	Brian Bolland
The Man of Steel	3		Superman	John Byrne	John Byrne
Spiderman: Chapter One	3		Spiderman	John Byrne	John Byrne
Untold Stories of Spiderman	21		Spiderman	Kurt Busiek	Pat Oliffe
Sandman	73		Sandman	Neil Gaiman	Michael Zulli
Star Wars: Shadows of The Empire	2		Star Wars	John Wagner	Kilian Plunkett
Star Trek: The Trial of James T. Kirk	1		Star Trek	Peter David	James W. Fry
Fanhunter Classics	2		Fanhunter	Cels Pinyol	Cels Pinyol
Fanhunter Almanaque	1		Fanhunter	Cels Pinyol	Cels Pinyol

- **Trading Cards**

Título	Nº de Paquetes	Familia
Sandman		Sandman
Sin City		Sin City
X-Men Gold		X-Men
Star Wars Special Edition Deck		Star Wars

- **Pósters**

Título	Largo	Ancho	Dibujante	Familia
Pink Cadillac	55	36.9	Luis Royo	
A Million Tears	41	28.5	Luis Royo	
Star Wars: A New Hope Chrome Art Print				Star Wars
Buffy Limited Edition Lithograph	28	22	Julie Bell	

- **Camisetas**

Título	Talla	Tipo Manga	Familia
Seven of Nine T-Shirt	XL	Corta	Star Trek
Earth X	XL	Corta	
Sandman: The Wake	XL	Corta	Sandman
Boba Fett T-Shirt	L	Corta	Star Wars
Morsa	L	Corta	Fanhunter

- **Revistas**

Título	Número	Periodicidad	Familia
Babylon 5 Magazine	12	Bimensual	
X-Men Super Special	1		X-Men

- **Figuras**

Título	Ancho	Largo	Alto	Familia
Arabian Nights Sandman Mini Statue	0.75	0.75	5	Sandman
Star Wars: Luke and Leia Statue			10	Star Wars
Marv's Statue				Sin City
STAP with Battle Droid				Star Wars

- **Mugs**

Título	Familia
Death (by Bachalo)	Sandman
Batman Classic	Batman

- **Pins**

Título	Familia
300	
Don Depresor	Fanhunter
Morsa	Fanhunter
Star Wars: Rebel Alliance Logo Pin	Star Wars

- **Cintas de vídeo**

Título	Formato	Duración (min.)	Familia
Akira	PAL	124	
Dragon Ball Z: Episodes 1-25 (6tapes)	PAL	535	Dragon Ball
X-Men	NTSC	122	X-Men

- **Otros (zippos, bolígrafos, maquetas, caretas, ...)**

Título	Familia
Moe's Escort Service (zippo)	Sin City
Batman & Robin Wathc Set	Batman
Star Wars Trivial Pursuit	Star Wars

3.2. Consideraciones a tener en cuenta

El sistema debe deducir cierta información (mediante el uso de reglas en JESS) a partir de la disponible, de esta forma:

- El sistema frente a una compra de un producto debe:
  - Actualizar el histórico de preferencias de la tienda (clases PreTipo y PreFamilia), añadiendo una nueva instancia en Pretipo y/o Prefamilia o incrementando las unidades si ya existe esa preferencia por parte del mismo cliente hacia ese producto o familia.
  - Reducir el número de ejemplares disponibles del producto vendido y borrar dicha compra.
  - El sistema debe controlar que un determinado producto llega a cero ejemplares disponibles avisando a los empleados mediante algún tipo de mensaje.
- Frente a una actualización de preferencias del tipo que sea se debe estudiar la consolidación de dicha preferencia por parte del cliente:
  - Se considerará que un cliente consolida un gusto si existe una preferencia de dicho cliente de una familia o de un producto con un valor en el slot cantidad mayor que 3. Es decir el cliente ha comprado a lo largo del tiempo más de 3 unidades de dicha familia o dicho producto. En ese caso, esa familia o producto debe ser insertado en la lista de gustos correspondiente del cliente (si no está ya) y poner como definitiva dicha preferencia.

4. Trabajo a realizar de forma obligatoria

Introducción de las reglas necesarias en el sistema para poder controlar el proceso de inferencia comentado en el punto 3.2.

Para la evaluación de la práctica se deberá entregar por escrito el código de las reglas comentando el porqué de cada una de ellas, además se deberá entregar un disco con la implementación de dichas reglas.

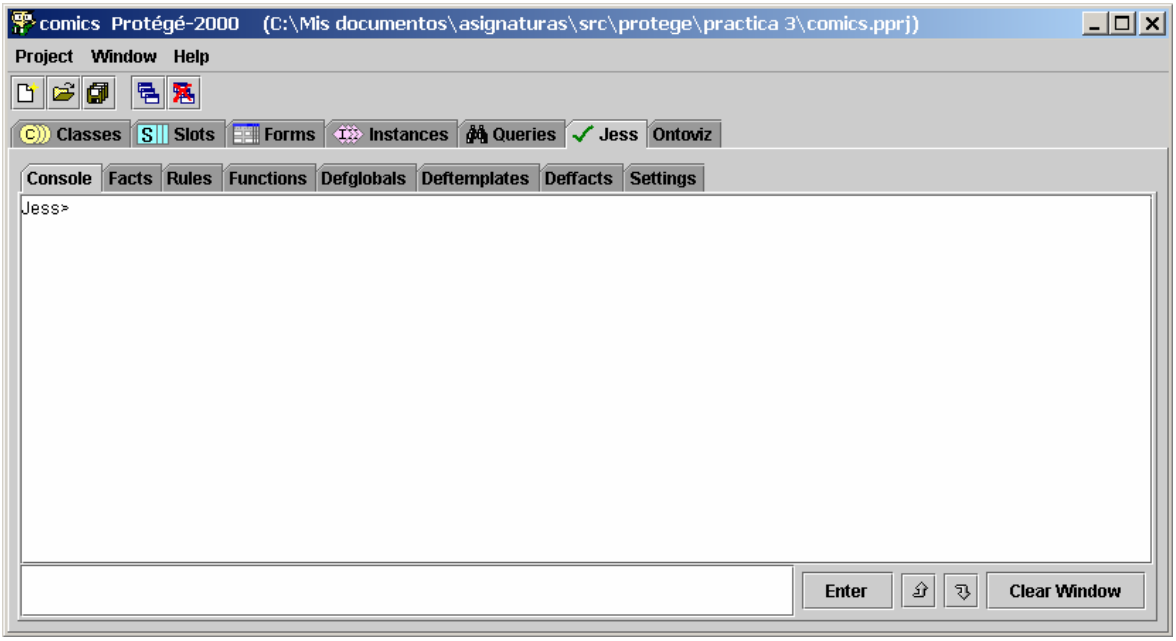
5. Posibles ampliaciones (opcional)

Añadir las reglas necesarias que permitan gestionar la realización de recomendaciones de productos existentes a los clientes frente a una compra empleando las listas de gustos de cada cliente.

ANEXO. Uso de *Jess* con Protege

Jess es un entorno de desarrollo de sistemas expertos mediante la generación de reglas de producción que puede ser empleado de forma conjunta con Protege con el objeto de inferir nuevo conocimiento a partir del disponible. Jess se integra con la herramienta Protege mediante una nueva carpeta o "tab".

Para activar la nueva carpeta si ésta no aparece consiste en, una vez abierto Protege, seleccionar dentro del menú "Project" la opción "Configure". Aparecerá una ventana donde se puede seleccionar que carpeta se quieren visualizar y cuales no. Hacer clic en la casilla donde aparece "JessTab" y pulsar "Ok". La carpeta Jess aparecerá tal y como se muestra en la siguiente figura.



Jess es ejecutado de forma interpretada a través de un *shell* donde se van introduciendo los comandos necesarios al mismo estilo que CLIPS. Al conjunto de comandos y funciones para crear reglas de CLIPS vistos en teoría se le añaden nuevos comandos para gestionar las clases, instancias y slots de PROTEGE. Los comandos más importantes a emplear son los que a continuación se muestran:

Básicos de funcionamiento

(batch nombre_fichero)

Carga los hechos y reglas especificados en el fichero de nombre "nombre_fichero".
Ej: (batch "w:\\practica3\\reglas.txt") Nota: El carácter "\" hay que ponerlo dos veces

(run)

Lanza el motor de inferencia, cada vez que se quieran probar las reglas habrá que ejecutar este comando.

(reset)

Vacía la lista de hechos inicializando el sistema de producción.

Integración con PROTEGE

(mapclass nombre_clase)

Crea hechos en Jess a partir de dicha clase de Protege y todos sus descendientes (clases e instancias). Si se hace un (mapclass :THING) se crean todos los hechos necesarios para trabajar con Jess. Es por ello que sería necesario ejecutar siempre como primer paso esta instrucción.

Un hecho de una instancia en la integración de Jess con Protege tiene fundamentalmente la siguiente forma:

(object (is-a *nombre_clase*) (OBJECT *dir_instancia*) (:DIRECT-TYPE *dir_clase*) (slot1 *valor*) ... (slotn *valor*))

donde: *nombre_clase* es la clase a la que pertenece la instancia
dir_instancia es la dirección interna de dicha instancia y es el valor que se emplea para referenciar dicha instancia desde otras instancias
dir_clase es la dirección interna de la clase a la que pertenece la instancia

Un hecho de una clase en la integración de Jess con Protege tiene fundamentalmente la siguiente forma:

(object (OBJECT *dir_clase*) (:DIRECT-SUPERCLASSES *lis_dir_clase1*) (:DIRECT-SUBCLASSES *lis_dir_clase2*) …)

donde: *dir_clase* es la dirección de la clase en cuestión y es el valor que se emplea para referenciar dicha clase desde otras clases
lis_*dir*_clase1 es la lista de las direcciones de las superclases (clases padre)
lis_*dir*_clase2 es la lista de las direcciones de las subclases (clases hijo)

(make-instance [<instance-name>] of <class-name> <slot-override>* [map])

Crea una instancia de nombre "instance_name" (es opcional ponerle nombre) en la clase "class_name". Es posible crear la instancia dando valores a sus slots poniendo el nombre

del slot y posteriormente su valor. La opción map es para asegurarse que se crea el hecho correspondiente.
Ej.

(make-instance P1 of Persona Nombre Pepe Apellido Perez map)

(modify-instance <instance-name> <slot-override>*)

Permite modificar una instancia cambiando los valores de sus slots. Ej:

(modify-instance P1 of Persona Apellido Lopez)

(unmake-instance <instance-expression>+)

Esta función permite borrar una instancia indentificada por su dirección de instancia <instance-expression>.

(slot-get <instance> <slot-name>)

Esta función permite obtener el valor a un slot de una instancia identificada con su dirección en <instance>. Ej: (se supone que la vble. ?i1 apunta a una instancia de persona)

(slot-get ?i1 Nombre) → devolvería el nombre de la persona correspondiente

(slot-set <instance> <slot-name> <new-value>)

Esta función permite dar el valor a un slot de una instancia identificada con su dirección en <instance>. Ej: (se supone que la vble. ?i1 apunta a una instancia de persona)

(slot-set ?i1 Nombre Juan) → Pondría el valor Juan como nombre de la persona correspondiente

(slot-insert$ <instance-expression> <mv-slot-name> <index> <expression>+)

Esta function inseta un valor en un slot múltiple de una instancia identificada con su dirección en <instance-expression>. Se debe indicar en <index> la posición a ocupar en la lista por el valor, por defecto poner un 1. Ej: (se supone que la vble. ?i1 apunta a una instancia de persona y que dispone de un slot "teléfono" que es múltiple)

(slot-insert$?i1 teléfono 1 "963445211")

(slot-delete$ <instance-expression> <mv-slot-name> <range-begin> <range-end>)

Permite borrar valores de un slot múltiple de una instancia identificada con su dirección en <instance-expression>. Se debe indicar en <range-begin> y en <range-end> el rango de valores a borrar. Ej: (suponemos lo mismo que en el ejemplo anterior)

(slot-delete$?i1 teléfono 1 3) → borraría del elemento 1 al 3 del slot telefono

(facet-get <class-name> <slot-name> <facet-name>)

Obtiene el valor de la faceta <facet-name> del slot <slot-name> de una clase <class-name>. Ej:

(facer-get Persona Nombre :SLOT-DEFAULTS)

Valores posibles: :NAME, :DOCUMENTATION, :SLOT-DEFAULTS, :SLOT-MAXIMUM-CARDINALITY, :SLOT-MINIMUM-CARDINALITY, :SLOT-NUMERIC-MAXIMUM, :SLOT-NUMERIC-MINIMUM, :SLOT-VALUE-TYPE, :SLOT-VALUES, :SLOT-INVERSE, :SLOT-CONSTRAINTS

(facet-set <class-name> <slot-name> <facet-name> <new-value>)

Cambia el valor de la faceta <facet-name> del slot <slot-name> de una clase <class-name>. Ej:

(facer-set Persona Nombre :SLOT-DEFAULTS “Juan”)

Valores posibles: :NAME, :DOCUMENTATION, :SLOT-DEFAULTS, :SLOT-MAXIMUM-CARDINALITY, :SLOT-MINIMUM-CARDINALITY, :SLOT-NUMERIC-MAXIMUM, :SLOT-NUMERIC-MINIMUM, :SLOT-VALUE-TYPE, :SLOT-VALUES, :SLOT-INVERSE, :SLOT-CONSTRAINTS

(superclassp <class1-name> <class2-name>)

Determina si una clase <class1-name> es superclase de otra clase <class2-name>. Devuelve TRUE si se cumple y FALSE en caso contrario.

(subclassp <class1-name> <class2-name>)

Determina si una clase <class1-name> es subclase de otra clase <class2-name>. Devuelve TRUE si se cumple y FALSE en caso contrario.

Existen muchas más funciones que pueden consultadas en el documento suministrado en la página web de la asignatura.

EJEMPLO de regla empleando JESS y PROTEGE sobre el sistema de frames de la práctica:

```
;; Qué ha comprado quién

(defrule prueba1
  (object (is-a Compra) (Cli ?c) (Prod ?p) (Unidades ?u)  )
  (object (is-a Cliente) (OBJECT ?c) (Nombre ?n) )
  (object (is-a ?Cl) (OBJECT ?p) )
  =>
  (printout t ?n " ha comprado ahora un/a " ?Cl crlf)
)
```

Esta regla se dispara si hay un hecho Compra (es decir hay una instancia de Compra) y buscamos el nombre del cliente y el tipo de producto comprado.

El disparo provoca la salida por pantalla de un mensaje como por ejemplo:

Pepe ha comprado ahora un/a Revista

Si la compra realizada fuese hecha por Pepe y hubiese comprado una revista.