INTELIGENCIA ARTIFICIAL

Practica de SBCs

Enero 2007

MTI

Albert Gómez Companys

Iván Fernández Zwanziger

INDICE

1. Introducción

En esta práctica intentaremos elaborar un sistema experto, utilizando los conocimientos de ingeniería del conocimiento vistos en las clases de teoría. Para ello, primero analizaremos el problema y veremos los requerimientos del enunciado, luego seguiremos con la construcción de una ontología completa para nuestro sistema con Protègè, y finalmente elaboraremos el sistema experto con la herramienta CLIPS.

2. Identificación del problema

El problema consiste en la elaboración de un programa para diseñar menús para una empresa de catering.

La función del sistema, tendrá que ser que dadas unas preferencias del usuario y unas restricciones, el sistema tiene que ser capaz de buscar en su conocimiento (ontología), y a partir de ello elaborar tres menús diferentes, que estén en una franja de precios dada por el usuario.

Vamos a ver qué tipo de preferencias o requerimientos tiene que dar el usuario al sistema:

- Tipo de evento:

Si es un evento familiar, un congreso… No vamos a darle mucha importancia a este requerimiento ya que los menús no variarán mucho por este dato.

- Época del año:

Tendremos que saber en qué época del año estamos, ya que quizás algunos ingredientes solo estén disponibles en algunas épocas, o por ejemplo, las sopas no serían muy adecuadas para el verano.

- Número de comensales:

Si la comida tiene que ser para mucha gente, intentaremos evitar platos muy complejos, o por ejemplo intentaremos que los primeros platos sean fríos.

- Intervalo de precios:

Precio máximo y mínimo que tiene pensado el usuario en gastarse.

- Tipo de comida:

Si la comida tiene que ser vegetariana, si hay algún elemento que esté prohibido (por ser alguien alérgico…). Habrá que tener en cuenta que hay unos platos limitados, y que si elegimos una comida vegetariana, tendremos muchas menos opciones y quizás los menús serán muy

parecidos. Lo mismo pasaría si marcamos *cebolla* como ingrediente prohibido, ya que muchos platos lo usan para su elaboración.

- Vino:

En la ontología dispondremos de una carta de vinos, y el usuario podrá elegir si quiere un vino para cada plato, o si sólo quiere uno. En el caso de escoger un vino para cada plato, el sistema tendrá por defecto un vino recomendado por la casa, que se adecuará tanto al tipo de plato (si es pescado será vino blanco), como al precio (si el plato es caro, el vino será más bueno).

- Estilo de comida:

Tendremos 4 tipos de comida: Tradicional, moderno, sibarita y regional.

La comida tradicional serán platos más típicos. También serán los más baratos.

Si elegimos cocina moderna, los platos tendrán raciones más pequeñas, pero serán platos más sofisticados y por tanto también más caros.

Para los menús sibaritas o gourmet, los platos serán de máxima calidad, con platos exclusivos y con un precio muy elevado.

Y por último, pondremos algún ejemplo de comida regional, que tendrá un precio normal.

Hay que tener en cuenta que al introducir las restricciones o preferencias hay que ser un poco consistente en las decisiones. Es decir, no podemos pretender tener un menú de gourmet, con un presupuesto bajo, o si queremos vino para cada plato también nos saldrá más caro…

3. Conceptualización y Formalización

En este apartado explicaremos todos los pasos para la construcción de la ontología. Para ello, utilizaremos la guía de Ontología 101, ya que después de haberla leído, creemos que es una buena base para comenzar a crear una ontología desde cero.

3.1. Determinar el dominio y alcance de la ontología

Para poder determinar el dominio de la ontología, Ontology 101 nos propone hacernos una serie de preguntas que deberemos responder:

- ¿Cuál es el dominio que la ontología cubrirá?

La ontología debe cubrir una amplia gama de platos, teniendo en cuenta que tenemos que tener diferentes primeros platos, segundos platos y postres, y que por cada tipo de plato, deberemos tener diferentes opciones de estilo de cocina (tradicional…). Además, también deberá cubrir una gama de vinos para poderlos relacionar con los diferentes platos.

- ¿Para qué usaremos la ontología?

Para elaborar menús de una empresa de catering. Los menús deberán incluir un primer plato, un segundo plato y un postre, y según los requerimientos del usuario, un vino para toda la comida, un vino para cada plato, o simplemente refrescos.

-¿Para qué tipo de preguntas la información en la ontología deberá proveer respuestas?

Está claro que no vamos a poder poner todas las posibles preguntas, pero poner algunas nos puede ayudar en el posterior desarrollo de la ontología. Por ejemplo:

¿Qué primer plato de estilo tradicional no contiene zanahoria?

¿Existe un segundo plato de estilo gourmet con un precio inferior a 30 euros?

¿Podemos confeccionar un menú tradicional con un precio mínimo de 300 euros por persona?

¿Disponemos de algún primer plato típico de Francia?

¿La paletilla de cordero es primer plato o segundo plato? ¿Es un plato caliente o frío?

¿Qué vino recomienda la casa para un salpicón de marisco?

- ¿Quién usará y mantendrá la ontología?

Como ya hemos dicho, la ontología va dirigida a una empresa de catering, la cual tendrá que tener actualizada su ontología segundo vaya añadiendo o quitando platos de su carta.

3.2. Considerar la reutilización de ontologías existentes

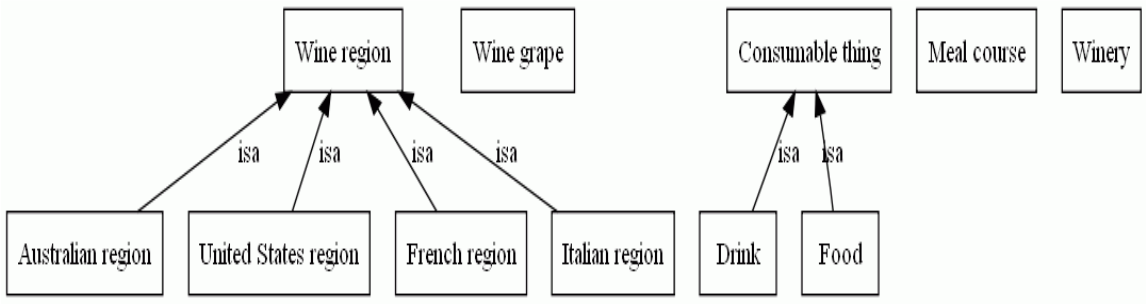
Uno de los aspectos en que las ontologías tienen ventajas, es en la posible reutilización de las mismas. Es decir, que si nosotros estamos pensando en crear una ontología, quizás alguien haya pensado hacer lo mismo anteriormente, y por tanto, es absurdo hacer lo mismo dos veces. Por eso, existen repositorios de ontologías disponibles para que cualquiera que quiera diseñar una ontología, pueda consultar previamente si tiene información útil.

Evidentemente, será difícil que encontremos una ontología exacta para nuestros requerimientos, pero otra ventaja de las ontologías es que se pueden modificar y adaptar fácilmente.

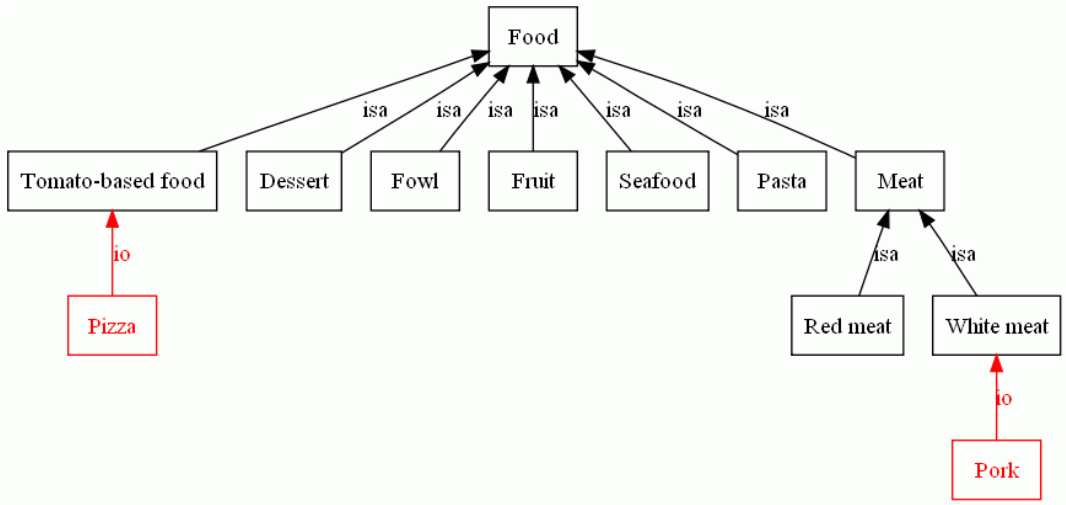
Para nuestro caso, hemos encontrado una ontología de vinos y alimentos, elaborada por ***Deborah McGuiness*** del *Laboratorio de Sistemas del Conocimiento* de la *Universidad de Standford* (universidad creadora de Protègè).

Esta ontología contiene una gran grama de vinos según su región, y además tiene una clasificación de los alimentos para poder crear relaciones entre los vinos y los alimentos.

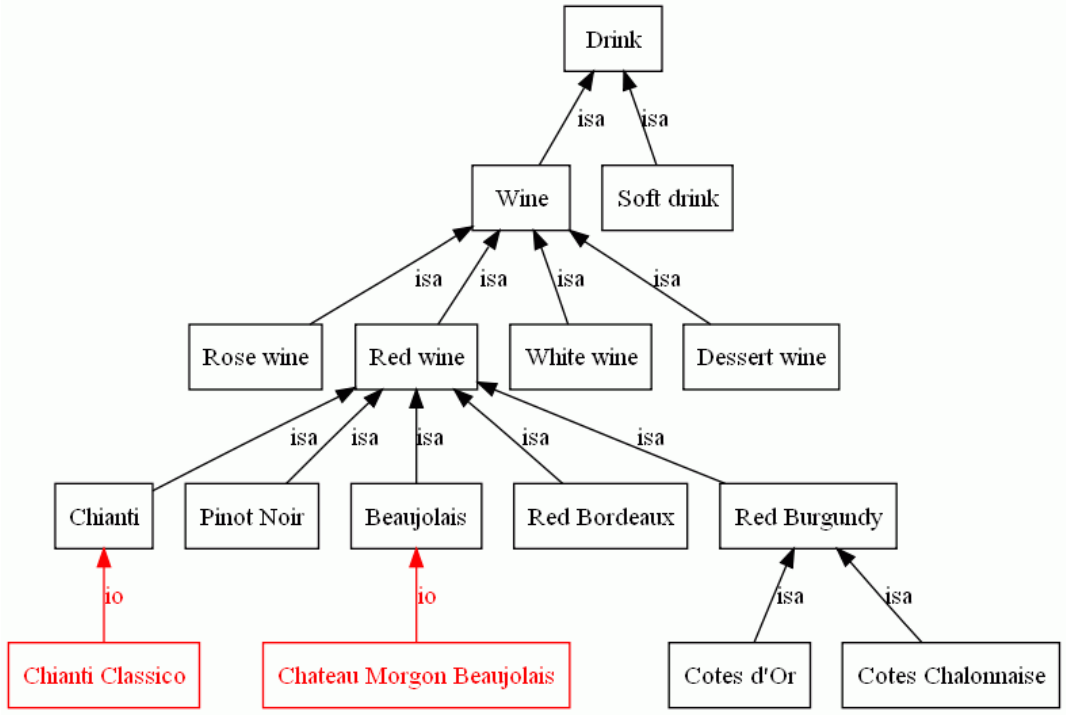
Vamos a ver un poco los esquemas básicos de la ontología. Los vamos a descomponer en diferentes grafos ya que la visualización de la ontología completa es complicada debido a su tamaño:



No haremos mucho caso a la clase *Wine Region*, ya que simplemente contiene información sobre diferentes regiones dónde se elabora el vino. Vamos a explorar las clases *Drink* y *Food*:



Este es un ejemplo de la clasificación de las comidas, y algunas de sus instancias. Vemos que *Meat* es un tipo de comida, y que podemos tener carne blanca y carne roja. Cerdo es una instancia de carne blanca.



Aquí vemos una pequeña clasificación de vinos, los tipos de vinos (rosado, tinto, blanco y postre) y diferentes denominaciones para cada tipo de vino.

3.3. Enumerar términos importantes para la ontología

Antes de comenzar con la creación de la ontología, es útil marcar los términos importantes que debemos incluir en la ontología, sin tener que pensar si eso serán clases, o si van a ser slots. Podríamos decir que es equivalente a hacer una lluvia de ideas, pero siempre relacionadas con el dominio que hemos delimitado previamente.

Así pues, algunos de los términos útiles para nuestra ontología pueden ser: *precio, primer plato, carne, pescado, marisco, comida tradicional, gourmet, comida regional, segundo plato, postre, comida caliente, vino, vino compatible…*

3.4. Definir clases y jerarquía de clases

Hay diferentes tipos de estrategias para elaborar la jerarquía. Si comenzamos por las clases más genéricas y acabamos por las más específicas, utilizaremos el proceso ***top-down***. En cambio, si comenzamos por las más específicas y vamos subiendo hasta la más general, habremos utilizado el procedimiento ***bottom-up***. Y por último, hay el proceso ***combinado***, que es una combinación de las dos.

En nuestro caso, diremos que hemos utilizado un proceso combinado, ya que como hemos comenzado a partir de una ontología ya creado, la hemos ido adaptando poco a poco hasta llegar a una ontología que se adapte a todas nuestras necesidades.

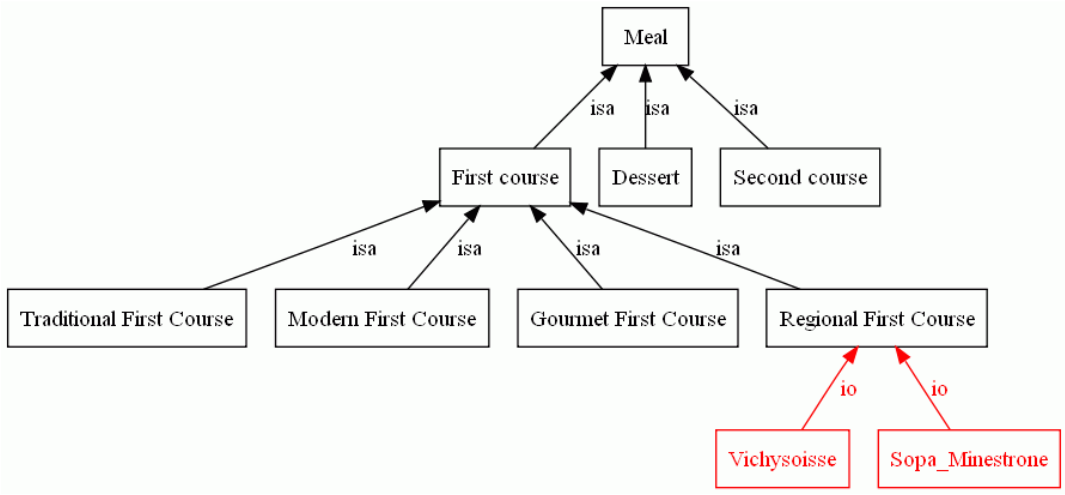
Partiendo de la ontología previa, hemos decidido que no vamos a modificar nada de la clase Drink. Quizás haya muchas cosas que no sean necesarias como por ejemplo saber de qué región se ha obtenido el vino, o el nivel de tanina, el cuerpo, etc. Pero consideramos que quizás un cliente que sea un gran conocedor de los vinos, le pueda interesar toda esa información, y aunque el dicho de “el saber no ocupa lugar” no sea exacto en el desarrollo de las ontologías, optaremos por dejar esa información.

Para los platos, una opción era clasificar todos los platos según el tipo de comida que sea (carne, pescado, ave…) o según si es primer plato, segundo plato, o postre. La opción más completa sería crear dos jerarquías, una para el tipo de comida y otro para el tipo de plato, ya que Protègè permite herencia múltiple, pero hemos decidido simplificar el problema y crear simplemente una jerarquía y añadir slots para la información restante.

Para tomar esa decisión, hemos pensado en como clasifican realmente los restaurante o empresas de catering sus comidas, y lo que hacen realmente es ofrecernos los platos según si es primer plato, segundo plato, etc. No nos preguntan si queremos carne, pescado… por lo que optaremos por hacer nosotros lo mismo, ya que además de ser una buena forma de clasificar los platos, será más útil i intuitiva para los usuarios finales.

Luego, tenemos que los platos pueden ser de tipo tradicional, moderno, gourmet… Eso también puede merecer una clasificación a parte. Podríamos ponerlo como slot, pero creemos que de esta forma estaría un poco más organizado.

Vamos a ver como quedaría la jerarquía de la clase comida:



Otra posible opción sería que el segundo nivel fuera el tipo de comida (tradicional…) en vez de el tipo de plato, es decir, invertir los niveles 2 y 3, ya que lo normal sería tener todo el menú del mismo tipo de cocina, pero pensamos que el término primer plato, es más global que el término cocina tradicional, por tanto lo hacemos así, y damos también la opción de que alguien quiera por ejemplo el primer plato y el segundo de cocina moderna, pero en cambio quiera un postre tradicional o típico.

3.5. Definir las propiedades de las clases: slots

Con la jerarquía que hemos creado no tenemos suficiente información para responder a todas las preguntas que necesita la ontología. Por tanto, todos los términos importantes que hemos destacado y no hemos incluido como clases serán seguramente slots.

El slot más general será el precio. Tenemos la clase *Consumable Thing*, de la cual cuelgan *Meal* y *Drink*. Sus hojas serán los platos o las bebidas, que van a tener todas un precio, por tanto, en la clase *Consumable Thing* podemos declarar un slot *Price*.

No vamos a fijarnos en los slots de *Drink*, ya que los hemos obtenido de la anterior ontología, y simplemente nosotros hemos añadido el precio.

Para la comida, hay diferente información que deseamos saber: El vino asociado a ese plato, si es caliente o no, los ingredientes que forman el plato, el tipo de comida (carne, pescado…) y por último en que temporada se puede consumir ese plato.

Todo ello, va a ser común para todas las subclases de la clase *Meal*, por tanto, pondremos ahí los slots para que se hereden en las subclases.

Con eso tenemos casi toda la información necesaria, simplemente tenemos que considerar el caso de los platos regionales. Deberemos saber de qué región es típico ese plato, por tanto añadiremos el slot *region*, que será un slot de la clase *Region*.

3.6. Definir las facetas de los slots

Una vez sabemos que slots queremos, tenemos que determinar los *facets* de cada slot. Vamos a ver pues que facets debe tener cada slot.

Consumable Thing:

- price: Debe ser de tipo Integer, y con cardinalidad single ya que un plato solo puede tener un precio.

Meal:

- associated wine: Será una instancia de Wine, y le pondremos cardinalidad single. Podríamos poner varios vinos asociados y por tanto cardinalidad múltiple, pero hemos decidido que la empresa tendrá simplemente un vino recomendado.

- hot: booleano que expresará si está caliente o no. Por defecto pondremos que estará caliente ya que la mayoría de platos lo son.

- ingredients: Instancia de la clase *Ingredient.* Será *required* ya que todos los platos tienen ingredientes, y le hemos puesto *multiple* ya que hay más de uno.

- season: Será de tipo *Symbol*, y aceptará los valores *ALL*, *WINTER* Y *SUMMER*. Por defecto tendrá el valor *ALL* ya que la mayoría de platos están disponibles en todas las épocas.

- type: Especifica el tipo de plato. También será de tipo Symbol pero especificaremos sus valores en las subclases.

First Course:

- type: Heredado de *Meal*, pero ahora permite los valores *VEGETAL*, *PASTA*, *SOUP*, *SEAFOOD*, *FISH*, *CREAM*.

Second Course:

- type: Heredado de *Meal*, pero ahora permite los valores *FISH, MEAT, FOWL, VEGETAL*. Por defecto *MEAL*.

Dessert:

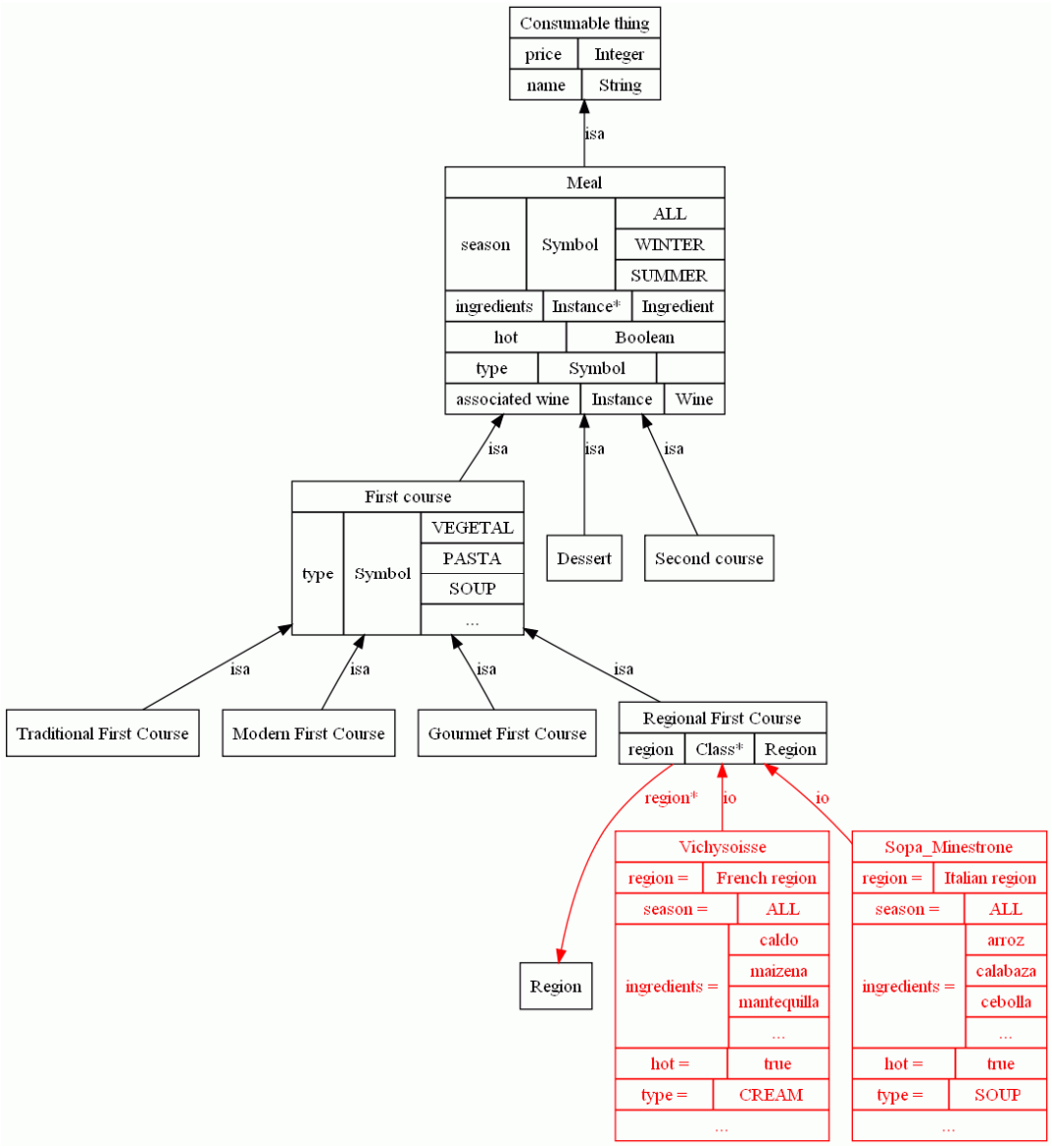
- associated wine: Heredado de *Meal*, pero ahora por defecto le vamos a asociar el vino de postres White hall Primavera.

- type: Heredado de *Meal*, pero ahora permite los valores *FRUIT*, *CAKE*, *ICECREAM*, *CREAM*.

Regional First Course, Regional Second Course, Regional Dessert:

- region: Clase con superclase Region.

Ahora vamos a ver ahora un grafo que muestre un ejemplo de la jerarquía hasta el primer plato de comida regional con sus slots y facets:



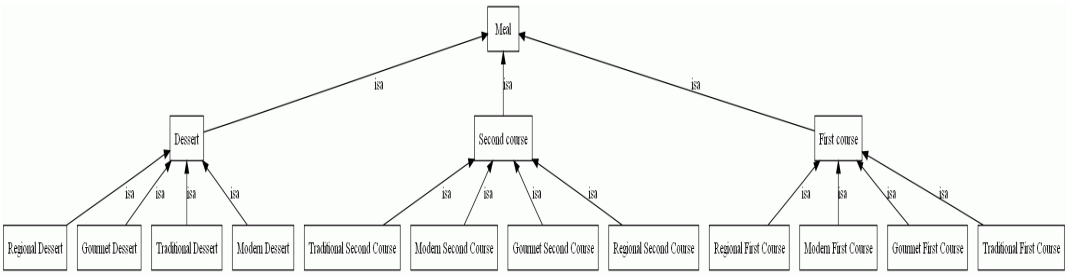
3.7. Crear instancias

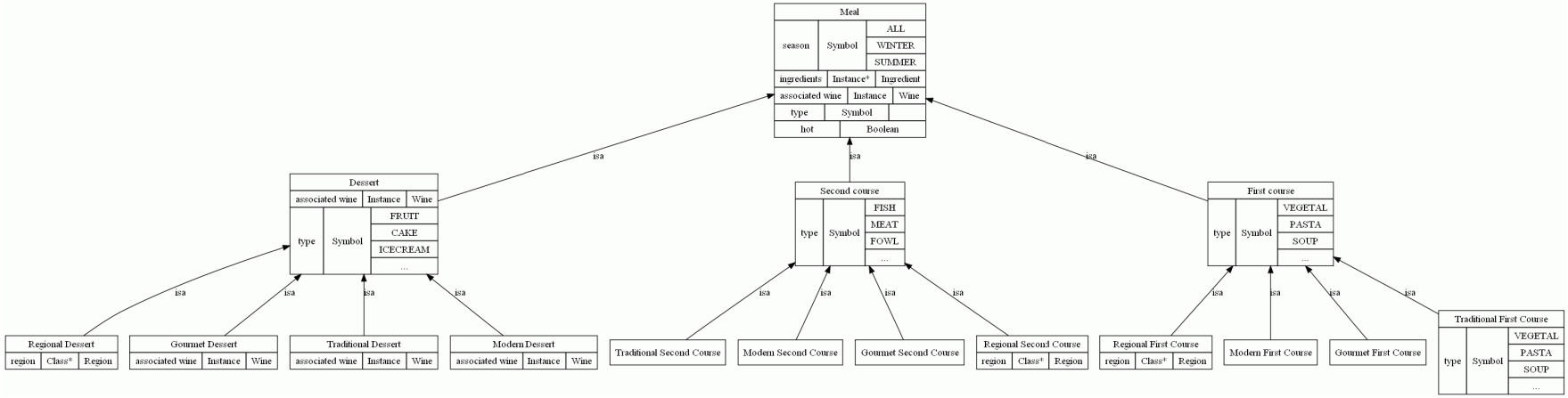
El último paso es crear todas las instancias necesarias para nuestra ontología. Para ello, hemos tenido que investigar en diferentes páginas webs de recetas y restaurantes de gran calidad para poder obtener algunos platos.

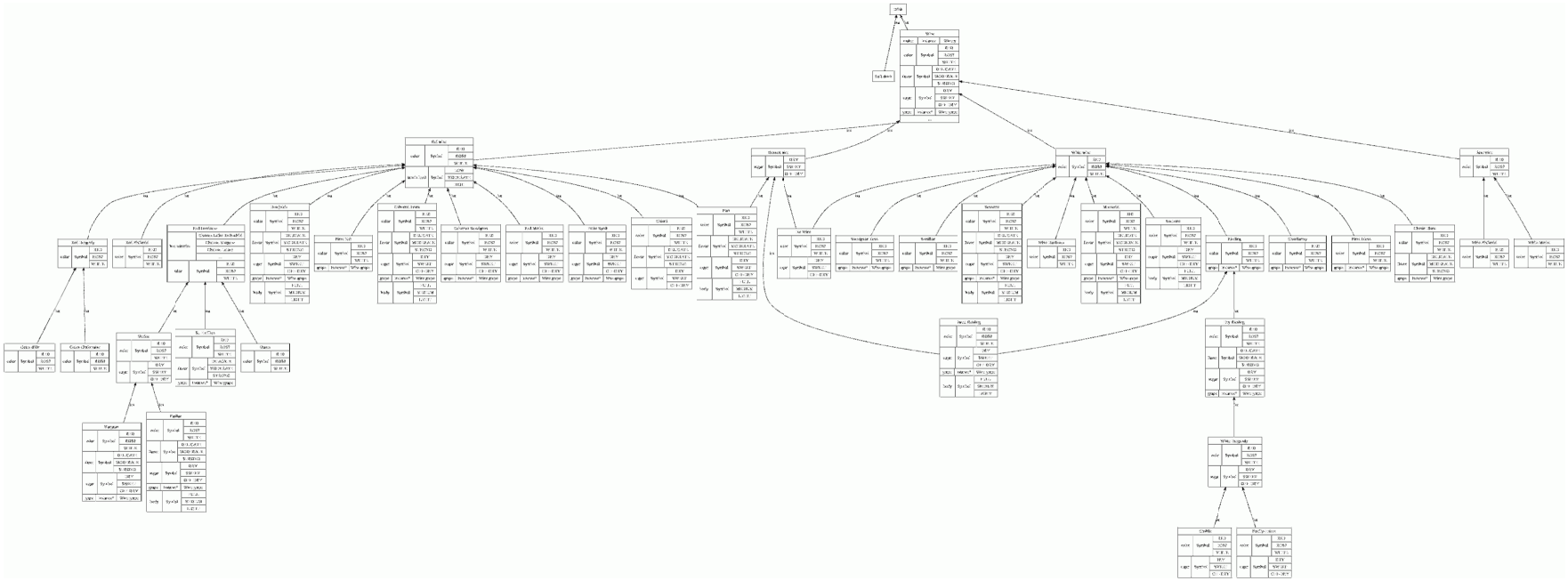
Este es seguramente el trabajo mas "fácil" en la elaboración de la ontología, pero seguramente también el más costoso en cuanto a tiempo. Nosotros hemos obtenido una ontología con unos 28 primeros platos, 34 segundos platos y 16 tipos diferentes de postre. Si bien no es todo lo completa que pudiéramos desear, ya que hay poca variedad de verduras o de comidas regionales, pensamos que para este problema tenemos más que suficiente, ya que realmente la oferta de los restaurantes o empresas de catering normalmente es bastante menor.

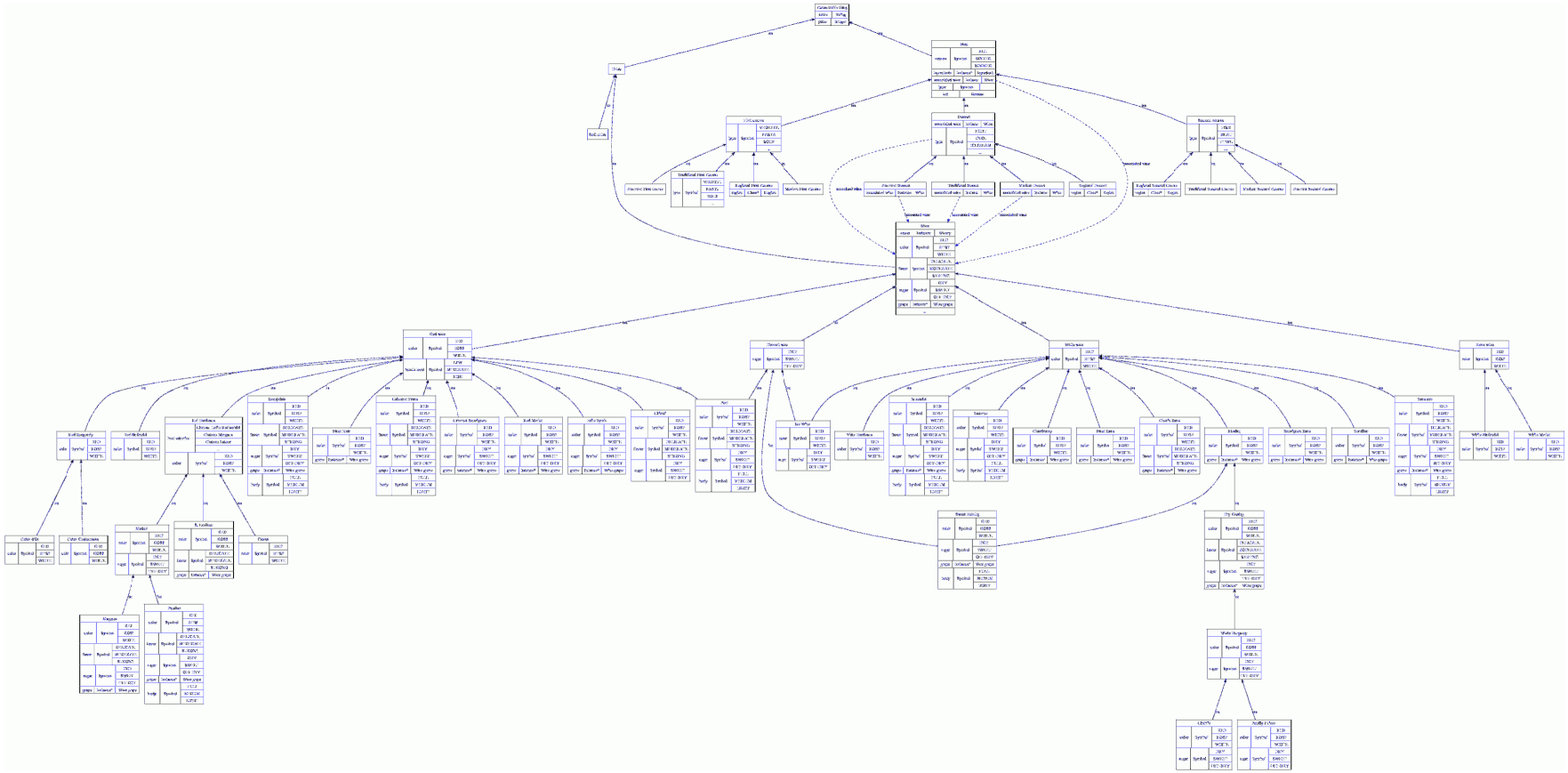
3.8. Ontología final

Finalmente vamos a mostrar unos grafos con la ontología final. Intentaremos mostrar la ontología por partes y finalmente toda entera, aunque su resolución no sea la óptima:




Meal
season
Symbol
ALL
WINTER
SUMMER
ingredients
Instance*
Ingredient
associated wine
Instance
Wine
type
Symbol
hot
Boolean
isa
Dessert
associated wine
Instance
Wine
type
Symbol
FRUIT
CAKE
ICECREAM
...
Second course
type
Symbol
FISH
MEAT
FOWL
...
First course
type
Symbol
VEGETAL
PASTA
SOUP
...
Regional Dessert
region
Class*
Region
Gourmet Dessert
associated wine
Instance
Wine
Traditional Dessert
associated wine
Instance
Wine
Modern Dessert
associated wine
Instance
Wine
Traditional Second Course
Modern Second Course
Gourmet Second Course
Regional Second Course
region
Class*
Region
Regional First Course
region
Class*
Region
Modern First Course
Gourmet First Course
Traditional First Course
type
Symbol
VEGETAL
PASTA
SOUP
...





4. Implementación

4.1. Modulo de inicialización del sistema

En este modulo simplemente tratamos de inicializar el sistema con los valores nulos o cero, como por ejemplo el presupuesto, los objetos temporales de recomendación y los datos del cliente.

```
(defmodule inicializacion
       (import MAIN ?ALL)
       (export ?ALL)
)

(defglobal ?*presupuesto* = 0)

(defrule init
       (not (object (is-a Menu)))
       =>
       (assert (recomendacion (nombre "recomen")))

       (bind ?ins (make-instance info of Datos))
       (assert (info-cliente (datos ?ins)))
)
```

4.2. Modulo de requisitos del usuario

En este modulo sirve para interactuar con el usuario. Concretamente vamos a hacerle varias preguntas para poder obtener sus requisitos y de esta forma poder seleccionar los menús acorde a sus preferencias.

Primero de todo vamos a preguntarle al usuario que tipo de comida quiere. Como ya hemos visto en la ontología, tenemos diferentes estilos de comida. Pueden ser comida tradicional, comida moderna, comida regional y comida para sibaritas.

Esta pregunta también está condicionado al precio final, ya que hemos diseñado que cada tipo de comida tiene una escala de precios acorde a su estilo. Esto quiere decir que la comida tradicional va a ser comida más sencilla, y por tanto más barata. En el otro extremo tenemos la comida sibarita, que tiene ingredientes exclusivos y que su elaboración es mucho más compleja, y consecuentemente será más cara.

En segundo lugar tenemos el presupuesto del usuario. Como ya hemos dicho tenemos que ser consecuentes con nuestra primera elección, y poner un valor de precio máximo acorde al estilo de comida. Si no hacemos esto no obtendremos ningún menú como resultado.

La condición de precio que hemos puesto para sacar los menús, es que el precio no pueda exceder del 20% del precio fijado por el usuario. No hemos puesto ninguna más restricción, ya que debido a que inicialmente ya restringimos la franja de precios con el estilo de comida, no es necesario y además nos podría inducir a errores y a no encontrar ningún menú.

Una vez tenemos estos valores generales, el usuario deberá elegir para cada plato que tipo de comida quiere. Por ejemplo, en el primer plato deberá elegir si quiere ensalada, sopa, marisco, pescado, etc. Las posibilidades de tipo de comida las tenemos restringidas en la ontología, ya que por ejemplo no es lógico que un primer plato sea de carne. Entonces para cada plato, le damos a elegir al usuario el conjunto de posibilidades definida por el slot de la ontología.

Después que el usuario haya seleccionado el tipo de comida para cada plato (primer plato, segundo plato y postre), se le dará a elegir si quiere vino.

En el caso que quiera vino, el sistema recomendara un vino para toda la comida. Esto lo haremos siguiendo lo ontología. Cada plato tiene definido un tipo de vino recomendado según el estilo y tipo de comida. Por ejemplo, si escogemos un estilo sibarita, y un primer plato que no sea pescado, vamos a obtener un vino tinto, y con un precio elevado. Entonces, para poder escoger un vino para toda la comida, simplemente escogeremos el vino recomendado del primer plato.

En el caso que quiera vino blanco, el usuario también podrá escoger esa opción. Y si el usuario no quiere ningún tipo de vino, entonces el sistema recomendara agua o algún refresco.

Creemos que estas preguntas nos dan una información básica sobre lo que quiere el usuario, y que simplemente con eso podemos darle unos menús que cumplan con las expectativas del usuario. Por otro lado, si el usuario nos da alguna otra restricción específica, siempre estamos a tiempo de mejorar el sistema añadiendo nuevas preguntas ya que tal y como hemos diseñado la ontología podemos añadir nuevas restricciones.

A continuación ponemos una parte de muestra de código de este modulo del sistema:

```
(defrule menu-a-crear
       ?rec <- (recomendacion (Menuslist $?lista-menus & : (= (length ?lista-
menus) 0)))
       =>
       (bind ?cjto-respuestas (create$ 0))
       (bind ?respuesta "")
       (printout t "¿Cual de los siguientes estilos de comida quiere para el
evento?" crlf)
       (printout t "    1. Clasico" crlf)
       (printout t "    2. Moderno" crlf)
       (printout t "    3. Regional" crlf)
       (printout t "    4. Sibarita" crlf)

       (printout t "Introduce el numero que indica el estilo que quiere: ")
       (bind ?respuesta (read))

       (switch ?respuesta
              (case 1 then
                     (bind ?ins (make-instance firstmenu of Menu))
                     (send ?ins put-nombre "Menu1")
                     (send ?ins put-estilo "Clasico")
                     (send ?ins put-precio 0)
                     (bind ?cjto-respuestas (insert$ ?cjto-respuestas 1
?respuesta))
                     (bind ?lista-menus (insert$ ?lista-menus 1 ?ins))

                     (assert (estado-menu (Menu ?ins) (estado pendiente)))
                     ;;segundo menu
                     (bind ?ins (make-instance secondmenu of Menu))
                     (send ?ins put-nombre "Menu2")
                     (send ?ins put-estilo "Clasico")
                     (send ?ins put-precio 0)
                     (bind ?cjto-respuestas (insert$ ?cjto-respuestas 1
?respuesta))
                     (bind ?lista-menus (insert$ ?lista-menus 2 ?ins))

                     (assert (estado-menu (Menu ?ins) (estado pendiente)))
                     ;;tercer menu
                     (bind ?ins (make-instance tercermenu of Menu))
                     (send ?ins put-nombre "Menu3")
                     (send ?ins put-estilo "Clasico")
                     (send ?ins put-precio 0)
                     (bind ?cjto-respuestas (insert$ ?cjto-respuestas 1
?respuesta))
                     (bind ?lista-menus (insert$ ?lista-menus 3 ?ins))

                     (assert (estado-menu (Menu ?ins) (estado pendiente)))
              )


...
```

4.3. Modulo de generación de menús

En este modulo se realiza el cálculo y generación los tres menús, en base a los requisitos del cliente tomados en el modulo anterior.

El sistema busca en la antología las instancias acorde a las restricciones de los slots, como tipo de comida, bebida. Se generan varias listas que contendrán primer plato, segundo plato, postre y bebidas que hayan cumplido las restricciones.

Si se encontraron platos y bebidas se empieza a generar los menús a recomendar. Una vez generado el menú con sus platos, la bebida y el precio se valida que cumpla con el presupuesto que el usuario dispone. Si una de las restricciones no se cumplen se descarta ese menú.

Al final del cálculo ya se contara con los tres menús recomendados para el usuario y se desplegarán en pantalla a detalle.

A continuación mostraremos partes del código de la generación del menú:

```
(defrule seleccionar-bebida-en-menu
        (declare (salience 10))
        (not (seleccion bebida ?))
        ?act <- (menu-actual (Menu ?me))
        ?aaprop <- (articulos-apropiados (Menu ?m2 & : (eq ?m2 ?me))(lista-bebida $?lista-
beb))
        =>
                (bind ?i 1)
                (if (<= ?i (length$ ?lista-beb))
                        then
                                (bind ?pre (send ?me get-precio))
                                (bind ?bebida (nth$ ?i ?lista-beb))
                                (send ?me put-bebida ?bebida)
                                (send ?me put-precio (+ (send ?bebida get-price) ?pre))

                                (if (> (length$ ?lista-beb) 1)
                                        then
                                                (unmake-instance ?bebida)
                                )
                )
             (assert (seleccion bebida ok))
)

(defrule validacion-menu-precio
        (declare (salience 10))
        (not (valida-precio-menu ?))
        ?act <- (menu-actual (Menu ?me))
        ?plato1 <- (seleccion plato1 ok)
        ?plato2 <- (seleccion plato2 ok)
        ?dessert <- (seleccion dessert ok)
        ?bebidas <- (seleccion bebida ok)
        ?rec <- (info-cliente (datos ?cl))
        =>
        (bind ?preciototal (send ?me get-precio))
        (bind ?presupuesto (send ?cl get-presupuesto))
        (if (> ?preciototal (* ?presupuesto 1.2))
          then
             (send ?me put-precio 0)
               (retract ?plato1)
               (retract ?plato2)
               (retract ?dessert)
               (retract ?bebidas)
        )
     (assert (valida-precio-menu ok))
```

4.4. Pruebas

Por último vamos a mostrar algunos ejemplos de pruebas que hemos hecho para mostrar los resultados finales del programa.

Ejemplo 1:

Restricciones:

Estilo de comida: Sibarita
Presupuesto: 135
Primer plato: Pescado
Segundo plato: Carne
Postre: Crema
Bebida: Vino tinto

```
LISTA DE MENUS RESULTANTES

Nombre: Menu1
---------------------------------------
Precio: 131
Estilo: Sibarita
bebida: MAIN::Marietta+Zinfandel
Platos:
--------------------
   MAIN::Soufle_de_Chocolate
   MAIN::Tartar_de_Ternera_al_Perfume_de_Calvados
   MAIN::Humus_con_Ventrisca_de_Bonito
=======================================

Nombre: Menu2
---------------------------------------
Precio: 135
Estilo: Sibarita
bebida: MAIN::Elyse+Zinfandel
Platos:
--------------------
   MAIN::Soufle_de_Chocolate
   MAIN::Paletilla_de_Cabrito_al_Horno_con_Patatas_Panaderas
   MAIN::Humus_con_Ventrisca_de_Bonito
=======================================

Nombre: Menu3
---------------------------------------
Precio: 140
Estilo: Sibarita
bebida: MAIN::Chateau+Morgon+Beaujolais
Platos:
--------------------
   MAIN::Soufle_de_Chocolate
   MAIN::Jarrete_de_Cordero_Relleno_de_Datiles_con_Salsa_de_Cardamomo_Verde_y_Crema_de_Patatas
   MAIN::Blini_con_Caviar
=======================================
```

Podemos ver como tenemos tres menús diferentes y que están sobre la franja de precio indicada por el usuario.

Podemos observar como el postre siempre es el mismo, pero eso es porque en la ontología simplemente tenemos un postre que cumpla todas las restricciones impuestas.

Ejemplo 2:

Restricciones:

Estilo de comida: Tradicional
Presupuesto: 60
Primer plato: Sopa
Segundo plato: Ave
Postre: Helado
Bebida: Vino tinto

```
LISTA DE MENUS RESULTANTES

Nombre: Menu1
--------------------------------------
Precio: 50
Estilo: Clasico
bebida: MAIN::Marietta+Zinfandel
Platos:
--------------------
   MAIN::Helado_de_Vainilla
   MAIN::Pato_con_Jud%C3%ADas
   MAIN::Sopa_con_Salsichas
======================================

Nombre: Menu2
--------------------------------------
Precio: 65
Estilo: Clasico
bebida: MAIN::Elyse+Zinfandel
Platos:
--------------------
   MAIN::Helado_de_Vainilla
   MAIN::Faisan_con_Nata
   MAIN::Sopa_con_Salsichas
======================================

Nombre: Menu3
--------------------------------------
Precio: 51
Estilo: Clasico
bebida: MAIN::Chateau+Morgon+Beaujolais
Platos:
--------------------
   MAIN::Helado_de_Chocolate
   MAIN::Civet_de_Liebre
   MAIN::Consome
======================================
```

Los resultados de estas restricciones nos han dado unos menús que se parecen pero que son combinaciones de ellos. Por ejemplo, vemos que menu1 y menu2 comparten el primer plato. En cambio, el postre es diferente, pero entre menu2 y menu3, tenemos el mismo postre pero diferente primer plato.

Por otro lado, los segundos platos son todos diferentes, ya que tenemos más opciones donde elegir.

Por último, vemos otra vez que se cumple la escala de precios, con unos menús baratos de 51 y 50, y otro más caro de 65.

Ejemplo 3:

Restricciones:

Estilo de comida: Moderno
Presupuesto: 80
Primer plato: Pasta
Segundo plato: Pescado
Postre: Fruta
Bebida: Agua o refresco.

```
Nombre: Menu1
---------------------------------------
Precio: 67
Estilo: Moderno
bebida: MAIN::Limonada
Platos:
--------------------
   MAIN::Ananas_con_Sorpresa
   MAIN::Zarzuela_del_Mediterraneo_y_Bogavante
   MAIN::Risotto_de_Setas
=======================================

Nombre: Menu2
---------------------------------------
Precio: 63
Estilo: Moderno
bebida: MAIN::Coke
Platos:
--------------------
   MAIN::Ananas_con_Sorpresa
   MAIN::Lubina_a_la_Sal
   MAIN::Risotto_de_Setas
=======================================

Nombre: Menu3
---------------------------------------
Precio: 69
Estilo: Moderno
bebida: MAIN::Agua
Platos:
--------------------
   MAIN::Ananas_con_Sorpresa
   MAIN::Lenguado_a_la_Naranja
   MAIN::Raviolis_de_Foie_con_Salsa_Roquefort
=======================================

CLIPS>
```

Volvemos a apreciar tres menús diferentes que cumplen con los requisitos. Podemos ver cómo ha salido un tipo de bebida aleatoria, que puede ser agua, o cualquier otro tipo de refresco, en este caso limonada y coke. Los precios van de 63 el más barato, a 69 el más caro. Nosotros habíamos puesto un máximo de 80, pero al no escoger vino eso hace que el precio pueda ser más barato.

Ejemplo 4:

Restricciones:

Estilo de comida: Regional
Presupuesto: 60
Primer plato: Crema
Segundo plato: Carne
Postre: Pastel
Bebida: Agua o refresco.

```
LISTA DE MENUS RESULTANTES

Nombre: Menu1
--------------------------------------
Precio: 50
Estilo: Regional
bebida: MAIN::Limonada
Platos:
---------------------
   MAIN::Tiramis%C3%BA
   MAIN::Saltimbocca_de_Ternera
   MAIN::Vichysoisse
======================================

Nombre: Menu2
--------------------------------------
Precio: 50
Estilo: Regional
bebida: MAIN::Coke
Platos:
---------------------
   MAIN::Tiramis%C3%BA
   MAIN::Saltimbocca_de_Ternera
   MAIN::Vichysoisse
======================================

Nombre: Menu3
--------------------------------------
Precio: 49
Estilo: Regional
bebida: MAIN::Agua
Platos:
---------------------
   MAIN::St._Honore
   MAIN::Caracoles_a_la_Borgona
   MAIN::Vichysoisse
======================================
```

En este caso, como tenemos pocos platos regionales, vemos que hay dos menús que solo varia la bebida. Si ampliáramos la ontología con más información, podríamos pedirle al usuario de que zona quiere que sea típica la comida (por ejemplo el menú 3 sería un menú típico francés).

5. Conclusiones

Como conclusiones, podemos decir que hemos podido introducirnos en el mundo de las ontologías, y descubrir lo interesante que puede ser tener una buena ontología para crear sistemas basados en el conocimiento. Pensamos que esto es muy útil para poder crear sistemas que resuelvan pequeños problemas de la vida cotidiana, y eso lo hace más interesante ya que son los pequeños problemas los que afectan a más población.

Por otro lado, también hemos visto la importancia de poder compartir ontologías, ya que así no hace falta tener que reinventar algo que otra persona lo haya podido hacer previamente, y aunque no sea exactamente lo que necesitemos, nos puede ser útil para poder comenzar o para tener información adicional que podamos necesitar en un futuro.

En la parte de programación con CLIPS, hemos visto de qué forma podemos utilizar una ontología creada previamente, y el tener que usarla nos ha hecho también entenderla mejor. Por otro lado, se nos ha hecho un poco complejo el manejo de reglas con CLIPS, por ser un lenguaje nuevo al que nos hemos tenido que adaptar. Además, al estar acostumbrados a lenguajes procedurales, se nos ha hecho un poco más complejo este cambio, pero pensamos que al final lo hemos podido comprender bien y aunque nos hubiese gustado poder ampliar las restricciones y hacer un sistema más robusto, pensamos que hemos podido entender el concepto y el objetivo que tenia esta práctica.

Finalmente las pruebas han demostrado que el sistema funciona con las restricciones que le hemos impuesto, y aunque la ontología podía contener más platos, pensamos que para hacer las pruebas ya han sido suficientes.