Guión práctico del seminario 4 (26-11-2020)

1. Crearse un nueva ontología OWL con IRI: http://www.pizza.com/pizza.owl (no olvidar 'salvarla' en el disco duro con el nombre pizza.owl)
2. Ir a la pestaña 'Classes' y crearse las clases: Pizza, BasePizza, IngredientePizza
3. Hay que convertirlas en clases disjuntas (se puede hacer desde el menú Edit)
4. Seleccionar la clase BasePizza y desde el menú Tools, seleccionar 'Create class hierarchy':

    BaseAncha
    BaseDelgadaCrujiente
5. Hacer que las clases anteriores sean disjuntas
6. Seleccionar la clase IngredientePizza y con 'Create class hierarchy' , crear esta jerarquía de clases:

    IngredienteCarne
        IngredienteJamon
        Ingrediente Pepperoni
        IngredienteSalami
        IngredienteTernera
    IngredientePescado
        IngredienteAnchoas
        IngredienteAtun
        IngredienteGambas
    IngredienteQueso
        IngredienteMozzarella
        IngredienteParmesano
    IngredienteVegetal
        IngredienteAceitunas
        IngredienteAlcaparras
        IngredienteCebolla
        IngredienteChampiñones
        IngredientePimientos
            IngredienteJalapeño
            IngredienteRojo
            IngredienteVerde
        IngredienteTomate
7. Acordarse de hacer todos los 'siblings primitivos' disjuntos
8. Crearse las *propiedades de objetos*: cambiar a la pestaña : 'Object Properties'; crear esta jerarquía de propiedades
    tieneIngrediente
        tieneBase
        tienecubierta
9. Ahora hay que crearse las propiedades inversas, también desde 'Object Properties':
    esIngrediente
        esBase
        esCubierta

10. Asegurarse que las propiedades ‘tieneIngrediente’ y ‘esIngrediente’ quedan como propiedades ‘transitivas’ (si una propiedad s transitiva su inversa también, pero no puede ser funcional)
11. Hacer que la propiedad ‘tieneBase’ sea funcional
12. Ahora hay que especificar los *rangos* de las propiedades de objetos; asegurarse que la propiedad ‘tieneCubierta’ posee como rango ‘IngredientePizza’ y que la propiedad ‘tieneBase’ posee el rango BasePizza
13. También hay que especificar que ‘Pizza’ es el dominio de las propiedades anteriores
14. Comprobar que los rangos y los dominios están invertidos en las propiedades inversas
15. Definir restricciones existenciales para la clase ‘Pizza’ (se hace con el ‘Class expression Editor’):
    tieneBase some BasePizza
    tieneCubierta some IngredientePizza
16. Crearse subclases de ‘Pizza’ utilizando la propiedade de objetos ‘tieneCubierta’:
    PizzaNombrada
        PizzaMargarita (ingredientes: Mozzarella y Tomate)
        PizzaSoho (ingredientes: Mozzarella, Tomate, Parmesano y Aceitunas)
        PizzaAmericana (ingredientes: Mozzarella, Pepperoni y Tomate)
        PizzaAmericanaPicante (ingredientes: Mozzarella, Tomate, Pepperoni y Jalapeño)
17. Utilizar ‘Cloning’ desde el menú ‘Edit’ para crearse clases similares y hacer que las subclases de ‘PizzaNombrada’ sean todas disjuntas
18. Usar un razonador para obtener la *ontología inferida*
19. Introducir ingredientes inconsistentes en la jerarquía para determinar que el razonador los detecta y los muestra de color rojo
20. Crearse una subclase de Pizza: ‘PizzaConQueso’ que tenga al menos 1 cubierta que sea de los ingredientes de tipo queso
21. Convertir a la clase ‘PizzaConQueso’ en una *clase definida*
22. Utilizar el razonador para que ‘calcule’ (o infiera) automáticamente las subclases de ‘PizzaConQueso’
23. Condiciones *necesarias y suficientes*: crear una clase ‘PizzaVegetariana’ que tenga como cubierta sólo ingredientes de queso o vegetales
24. Utilizar el razonador y verificar la hipótesis de Mundo Abierto
25. Corregirlo definiendo clausuras de las propiedades de objetos de las clases ‘PizzaMargarita’, ‘PizzaSoho’, …
26. Volver a utilizar el razonador para comprobar que ahora si se detectan las pizzas vegetarianas

3-12-2020

27. Propiedades de tipos de datos:
    Cambiar a la pestaña : ‘Data Properties’ y crearse la propiedad : ‘tieneContenidoCalorificoValor’
    Cambiamos a la pestaña ‘Individuals’ y añadimos la instancia:
        MargaritaEjemplo
        En la caja Description seleccionamos ‘Types’ seleccionamos: ‘PizzaMargarita’
        En la caja Property asertions seleccionamos ‘Data property asertions’ y en la ventana de diálogo que se nos abrirá, seleccionar la propiedad

tieneContenidoCalorificoValor, después seleccionamos el tipo xsd:integer y tecleamos 263.

Hacemos lo mismo para crearnos las instancias: 'SohoEjemplo', 'AmericanaEjemplo', 'AmericanaPicanteEjemplo' a las que asignamos valores enteros: 310, 380 y 390 calorías

Hacemos lo mismo para crearnos la instancia: 'CuatroQuesosEjemplo' a la que asignamos el valor de 723 calorías

(17-12-2020)

28. Para poder hacer que todas las pizzas posean un *valor concreto de contenido calorífico*, hemos de establecer restricciones para las propiedades de tipos de datos:

    Nos cambiamos a la pestaña 'Clases'

    Seleccionamos la clase 'Pizza' y a través del botón '+' de la caja 'SubClassOf' se nos abrirá una ventana de diálogo donde seleccionamos la pestaña: 'Data restriction creator' y tecleamos:

    tieneContenidoCalorificoValor some xsd:integer ; lo que significa: *todas las pizzas han tener al menos un valor de contenido calorífico y este valor ha de ser un número entero.*

29. Crearnos clases de pizzas que especifican un rango de valores de calorías que nos interesan para realizar nuestra ontología, de acuerdo con la propiedad de tipos de datos 'tieneContenidoCalorificoValor'

    Seleccionamos la pestaña 'Clases' y nos creamos la subclase de Pizza: 'PizzaAltasCalorias'

    Seleccionamos 'Pizza' y el botón '+' de 'SubclassOf' y abrimos la ventana de diálogo del 'Class Expression Editor' y tecleamos:

    tieneContenidoCalorificoValor some xsd:integer [>= 400]

    Seleccionamos la pestaña 'Clases' y nos creamos la subclase de Pizza: 'PizzaBajasCalorias'

    Abrimos la ventana de diálogo del 'Class Expression Editor' y tecleamos: tieneContenidoCalorificoValor some xsd:integer [<400], de tal manera que no exista solapamiento con los miembros de 'PizzaAltasCalorias'.

30. Convertir 'PizzaAltasCalorias' y 'PizzaBajasCalorias' en clases definidas
31. Resincronizar el razonador y observar como en la ontología inferida aparecerá que la 'PizzaAltasCalorias' tiene como miembro 1 instancia de pizza (CuatroQuesosEjemplo) y la 'PizzaBajasCalorias' tiene 4 miembros (AmericanaEjemplo, AmericanaPicanteEjemplo, MargaritaEjemplo, SohoEjemplo).
32. Convertir a la propiedad de tipos de datos anteriormente declarada en 'Functional'
33. Crear una nueva clase de pizza a la que asignemos 2 valores de calorías con la propiedad 'tieneContenidoCalorificoValor'; resincronizar el razonar y comprobar que le ocurre a la jerarquía ontológica inferida

Practicando con la suposición de Mundo Abierto y sus consecuencias en la jerarquías ontológicas inferidas

34. Crearse una clase 'PizzaNoVegetariana'. Como queremos que esta clase sea una categoría de pizzas que contenga a todas las pizzas que no sean vegetarianas, hemos de crearnos una clase que sea complementaria con 'PizzaVegetariana'.

    Abrir en la ventana de diálogo de las subclases: 'Class expression editor' de la clase 'PizzaNoVegetariana' y teclear:

    Pizza and (not (PizzaVegetariana)))

Hay que añadir 'Pizza' a la condición necesaria y suficiente porque si no, estaríamos definiendo que cualquier individuo que no sea miembro de 'PizzaVegetariana' sería una 'PizzaNoVegetariana', sin importar si el individuo es pizza o cualquier otra cosa.

35. En la caja 'Description: PizzaNoVegetariana' declarar que es disjunta de 'PizzaVegetariana'
36. Convertir la clase 'PizzaNoVegetariana' es definida
37. Resincronizar el razonador y hemos de encontrar que para la nueva clase 'PizzaNoVegetariana' se han inferido 2 subclases: 'PizzaAmericana' y 'PizzaAmericanaPicante'
38. Crear una subclase de 'PizzaNombrada' que se llame 'UnClosedPizza'
39. Y seleccionarla, en la ventana de diálogo que se abrirá, teclear:
    'tieneCubierta' some 'IngredienteMozarrella'

    Luego, un individuo que sea miembro de 'UnClosedPizza' es necesario que sea también Miembro de 'PizzaNombrada' y posea una restricción existencial (*al menos uno*) para la relación 'tieneCubierta' con un individuo que sea miembro de la clase 'IngredienteMozzarella'. Debido a que no hemos añadido un axioma de cierre a la Propiedad 'tieneCubierta', un miembro de 'UnClosedPizza' puede poseer otros ingredientes aparte de 'IngredienteMozzarella'.
40. Resincronizar el razonador
    Podemos comprobar ahora que 'UnClosedPizza' no es ni 'PizzaVegetariana' ni 'PizzaNoVegetariana' y esto es debido a que según el *Razonamiento de Mundo Abierto* no existe un axioma de clausura definido para la relación 'tieneCubierta' y, por tanto, la pizza referida podría contener otros ingredientes aparte de Mozarella. Podríamos haber supuesto que 'UnClosedPizza' hubiera sido clasificada como 'PizzaNoVegetariana' ya que no lo ha sido como 'PizzaVegetariana', sin embargo, el *Razonamiento de Mundo Abierto* impide esa inferencia porque el hecho de que no se pueda determinar que 'UnClosedPizza' sea 'PizzaVegetariana' no implica que haya de ser 'PizzaNoVegetariana' (ya que también podría ser otra cosa). En conclusión, 'UnClosedPizza' no puede ser clasificada ni como 'PizzaVegetariana' ni como 'PizzaNoVegetariana'.

Creación de individuos . OWL nos permite crear individuos (similar a 'instancias' de las clases de los lenguajes OO) y afirmar propiedades sobre dichos individuos.

41. Crearse 'Pais' como subclase de 'Thing'
42. Cambiarse a la pestaña 'Individuals' y añadir la instancia: 'Italia'
43. En la caja 'Description' seleccionamos 'Types' y 'Pais' en la jerarquía de clases
44. Nos crearemos algunos individuos más, del tipo 'Pais', tales como: 'America', 'Inglaterra', 'Francia' y 'Alemania'
    Como OWL no utiliza la UNA ('unique name assumption'), se podría establecer que los individuos cumplan con las condiciones: 'SameAs', 'DifferentFrom' con respecto a otros individuos.

Restricciones 'hasValue' : definen conjuntos de individuos que poseen al menos una relación a través de una propiedad con 1 solo individuo específico

45. En la pestaña 'Object Properties' nos crearemos una nueva propiedad de objetos: 'tienePaisDeOrigen'
46. Seleccionamos 'IngredienteMozarella' en la jerarquía de clases, abrimos el 'Class expression editor' de esta clase y tecleamos:
    'tienePaisDeOrigen value Italia'

Al final, las condiciones que hemos especificado para el 'IngredienteMozzarella' han de afirmar que los miembros de la clase 'IngredienteMozzarella' son también miembros de la clase 'IngredienteQueso', están relacionados con el individuo 'Italia' a través de la relación 'tienePaisDeOrigen' y están relacionados con la clase 'Picante' a través de la relación 'gradoDePicante'

Clases enumeradas: OWL permite que las clases sean definidas mediante el listado preciso de sus individuos, que son miembros de una clase.

47. Seleccionar la clase 'Pais' dentro de la jerarquía de clases
48. Seleccionar la sección 'Equivalent Class' en la vista 'Description' , aparecerá una caja de texto donde teclearemos: {America, Inglaterra, Francia, Alemania, Italia} y aceptaremos

Los que nos ha aparecido significa que un individuo que sea miembro de la clase Pais debe ser uno de los individuos de la lista anterior, o de otra manera, la clase 'Pais' es equivalente a una nueva clase anónima que se define por enumeración.

49. Crear una nueva subclase denominada 'PizzaItaliana'
50. , abrimos el 'Class expression editor' de esta clase y tecleamos: 'Pizza and (tieneCubierta some (IngredientePizza and tienePaisDeOrigen value Italia))'
51. Convertiremos a 'PizzaItaliana' en una clase definida
52. Si sincronizamos el razonador, podemos comprobar que se infiere que 'PizzaItaliana' posee ahora 5 subclases: 'PizzaAmericana', 'PizzaSoho', 'PizzaMargarita', 'PizzaAmericanaPicante' y 'Unclosed Pizza'.