UNIVERSIDAD DE GRANADA

Desarrollo de Sistemas de Software basados en Componentes y Servicios

Creación de la Ontología de Libros en Protégé

Autor

Antonio José Muriel Gálvez

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

Granada, 23 de diciembre de 2024

Índice

Introducción

El objetivo de esta práctica fue diseñar una ontología en Protégé que representara información detallada sobre libros, autores, géneros y editoriales. El trabajo incluyó la creación de clases jerárquicas, propiedades de objetos y datos, restricciones, individuos de ejemplo, y la validación de consistencia mediante razonadores. En esta memoria se explica en detalle cada paso seguido en el desarrollo, los desafíos encontrados y cómo se resolvieron.

La ontología final sirve como una base para organizar datos sobre libros de manera lógica y estructurada, permitiendo consultas avanzadas e inferencias automáticas gracias a las capacidades de OWL (Web Ontology Language).

Configuración inicial

Creación de la ontología

- La ontología fue creada en Protégé como un archivo OWL denominado 'bookstore.owl'. El IRI asignado fue 'http://www.bookstore.com/ontology.owl', que funciona como un identificador único global para garantizar que los términos utilizados en la ontología no entren en conflicto con otros sistemas.
- Desde el menú principal de Protégé, se guardó el archivo en el disco local para evitar pérdida de información durante el desarrollo.

Configuración del entorno

Se habilitaron las vistas clave para trabajar con la ontología de manera eficiente:

- Clases: Para gestionar jerarquías conceptuales.
- Propiedades de Objetos (Object Properties): para definir relaciones entre clases.
- Propiedades de Datos (Data Properties): Para añadir atributos cuantitativos o cualitativos a las clases.
- Individuos (Individuals): Para crear ejemplos concretos dentro de la ontología.

Definición de clases y jerarquías

Creación de clases principales

Se definieron las clases fundamentales que forman el núcleo de la ontología:

- Libro: Representa los libros como entidad principal.
- Autor: Representa a los autores que escriben los libros.
- Género: Representa los géneros literarios.

- Editorial: Representa a las editoriales encargadas de publicar los libros.

Estas clases se crearon desde la pestaña **Classes** seleccionando la clase raíz ‘owl:Thing‘ y añadiendo subclases mediante clic derecho y la opción **Add Subclass**.

Creación de jerarquías

Dentro de las clases principales, se añadieron subclases para especificar categorías más detalladas: Subclases de ‘Género‘:

- Ficción
- NoFicción
- CienciaFicción
- Fantasía
- Biografía

Subclases de ‘Libro‘:

- LibroFísico: Libros impresos.
- Ebook: Libros en formato digital.

Para acelerar este proceso, se utilizó la herramienta **Create Class Hierarchy** disponible en Protégé.

Definición de clases disjuntas

Las subclases de ‘Género‘ y ‘Libro‘ fueron configuradas como disjuntas para evitar que un individuo pueda pertenecer a más de una subclase al mismo tiempo. Esto se logró seleccionando todas las subclases de una categoría y marcándolas como disjuntas en la sección **Disjoint Classes** del panel derecho.

Un libro no puede ser simultáneamente ‘Ficción‘ y ‘NoFicción‘. Esta restricción asegura la coherencia en la clasificación.

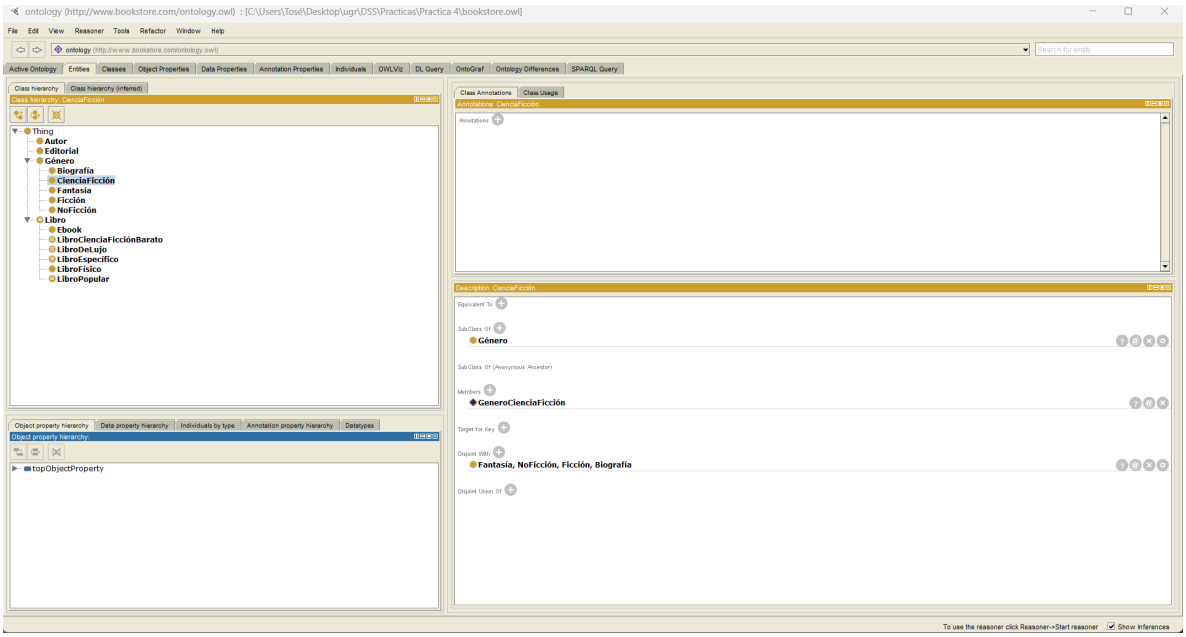

Figura 1: Clases Disjuntas

Propiedades de objetos e inversas

Definición de propiedades de objetos:

Las propiedades de objetos conectan clases entre sí. Se crearon las siguientes:

- escritoPor: Relaciona 'Libro' con 'Autor'.
- publicadoPor: Relaciona 'Libro' con 'Editorial'.
- tieneGénero: Relaciona 'Libro' con 'Género'.

Estas propiedades se añadieron desde la pestaña **Object Properties**, utilizando la opción **Add Object Property**.

Propiedades inversas

Se definieron propiedades inversas para facilitar la navegación bidireccional:

— 'escritoPor' 'haEscrito'.

— 'publicadoPor' $\iff$ '$esEditorialDe$'.

— 'tieneGénero' $\iff$ '$esGéneroDe$'.

Para crear una propiedad inversa, se seleccionó la propiedad original y se utilizó la opción **Add Inverse Property**.

Asignación de dominios y rangos

Se configuraron dominios y rangos para cada propiedad:

- ‘escritoPor‘:
  - Dominio: ‘Libro‘.
  - Rango: ‘Autor‘.
- ‘tieneGénero‘:
  - Dominio: ‘Libro‘.
  - Rango: ‘Género‘.
- ‘publicadoPor‘:
  - Dominio: ‘Libro‘.
  - Rango: ‘Editorial‘.

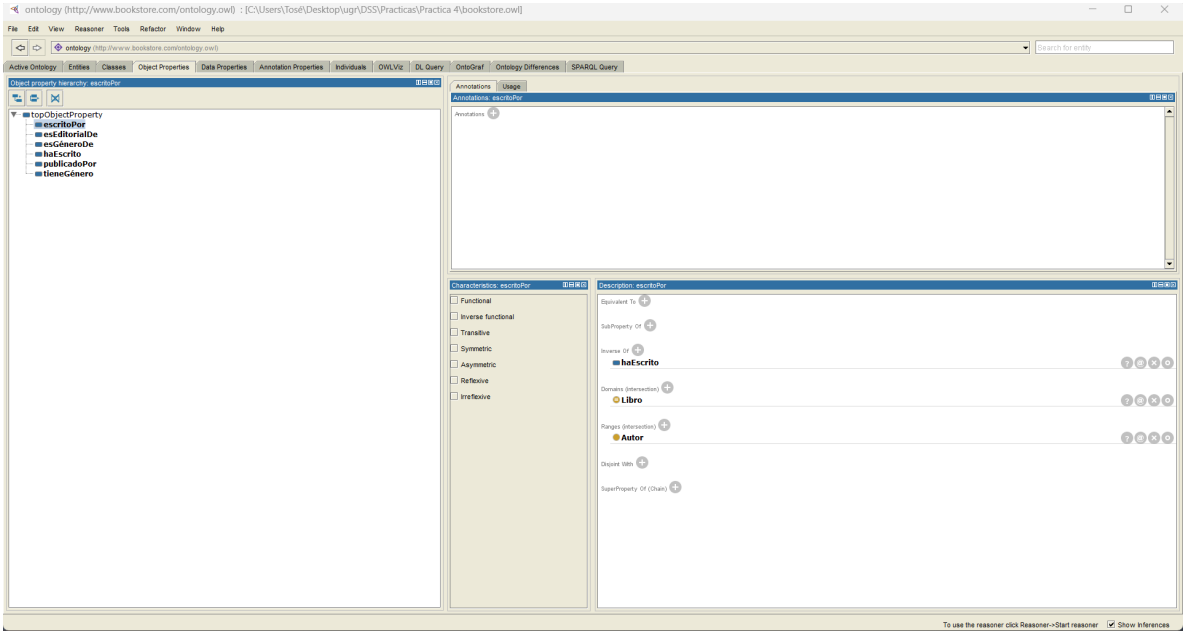

Figura 2: Object Properties

Restricciones y propiedades de datos

Restricciones existenciales

Se añadieron restricciones para la clase ‘Libro‘ utilizando el editor de expresiones:

- ‘escritoPor some Autor‘: Todo libro debe tener al menos un autor.
- ‘tieneGénero some Género‘: Todo libro debe pertenecer a un género.

Figura 3: Description Libro

Estas restricciones fueron configuradas en la sección **Class Description** de la clase 'Libro'.

Propiedades de datos

Se añadieron propiedades para capturar atributos adicionales:

- 'tienePrecio': Representa el precio del libro, con tipo de dato 'xsd:decimal'.
- 'tieneAñoPublicación': Representa el año de publicación, con tipo de dato 'xsd:integer'.

Creación de individuos

Se creó un individuo de ejemplo llamado 'EjemploLibro' con las siguientes características:

- Clase: 'LibroFísico'.
- Propiedades de datos:
  - tienePrecio = 29.99'.
  - 'tieneAñoPublicación = 2021'.

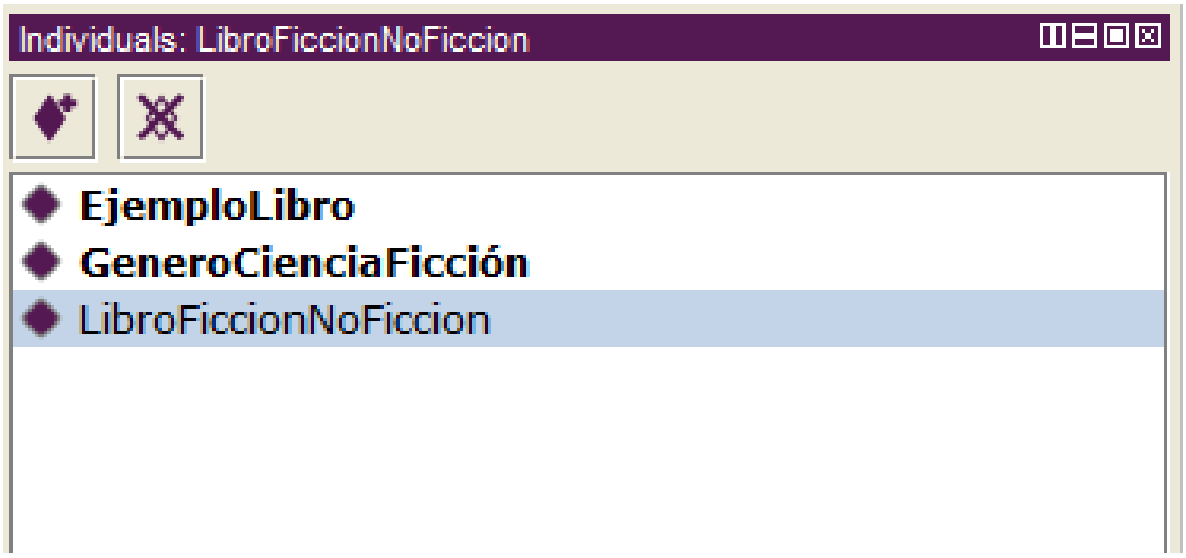

Figura 4: Individuals

Consultas avanzadas y razonamiento

Creación de la subclase ‘LibroCienciaFicciónBarato‘

Se definió una subclase de ‘Libro‘ con la siguiente restricción:

```
tieneGénero value GeneroCienciaFicción and tienePrecio some xsd:decimal [< 20]
```

Para implementar esta restricción, fue necesario crear un individuo ‘GeneroCienciaFicción‘ y asociarlo a la clase ‘CienciaFicción‘.

Prueba de consistencia

Se creó un individuo llamado ‘LibroInconsistente‘ que pertenece simultáneamente a ‘Ficción‘ y ‘NoFicción‘. Al ejecutar el razonador, Protégé detectó correctamente la inconsistencia debido a la disjunción entre estas clases.

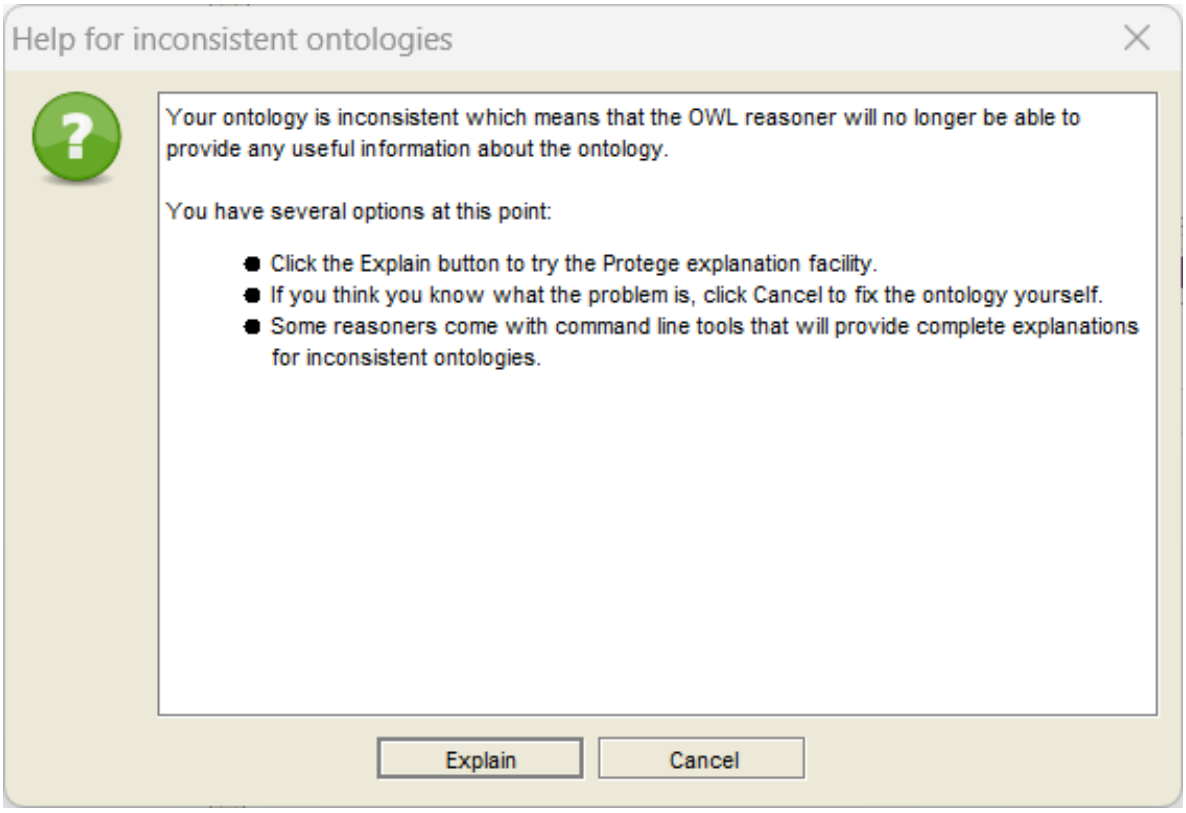

Figura 5: Error Raconador

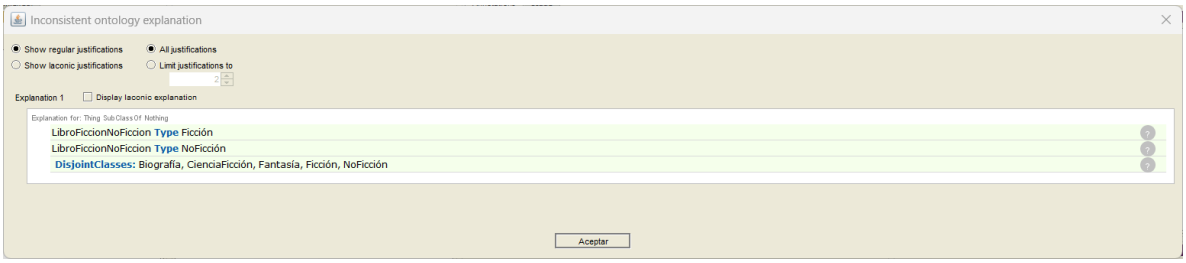

Figura 6: Error Raconador Explain

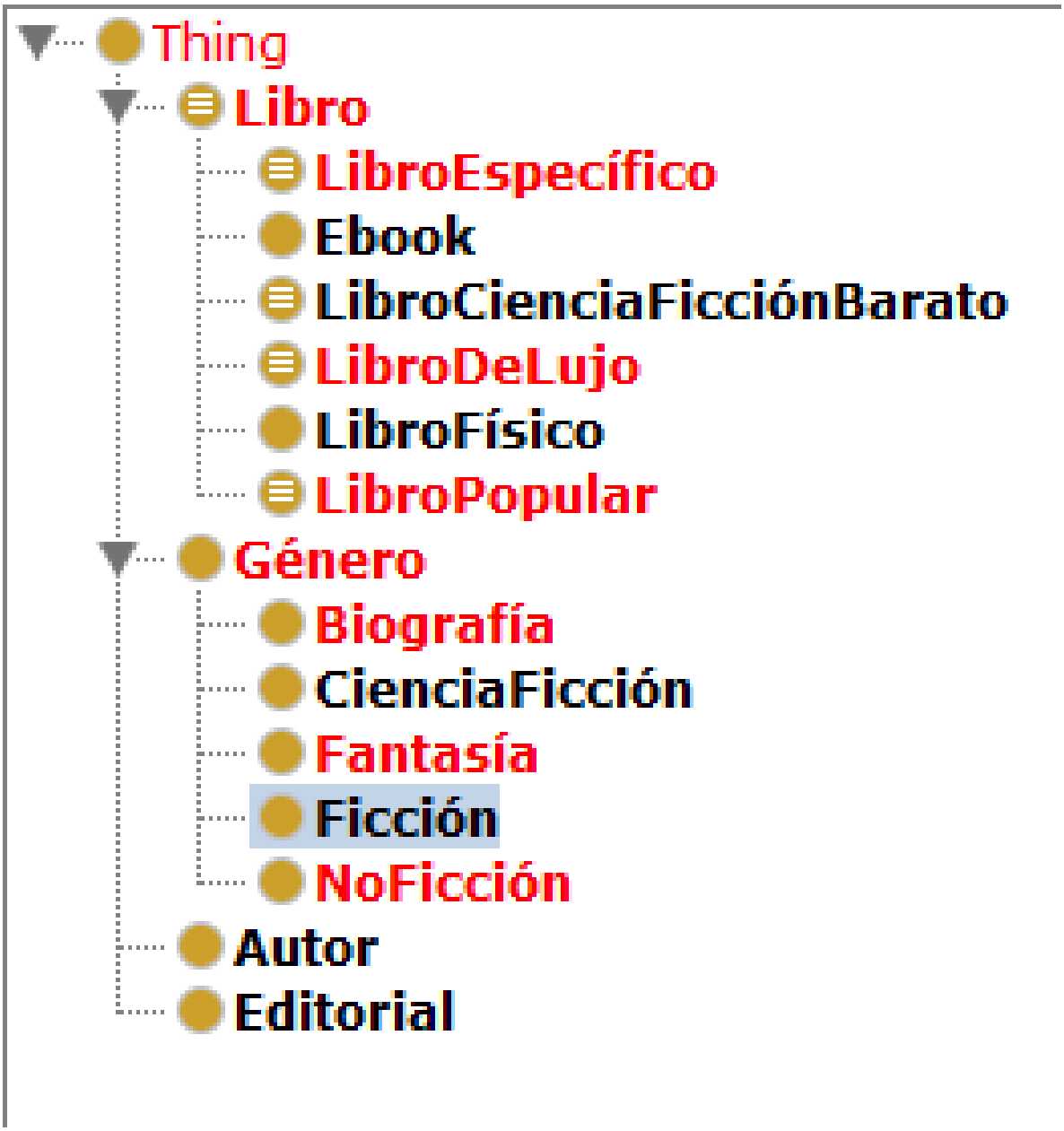

Figura 7: Error Entities

—

Conclusiones

La ontología final incluye una estructura jerárquica clara, propiedades definidas, restricciones lógicas y ejemplos de individuos.

Se verificó la consistencia utilizando un razonador, lo que garantiza su validez.

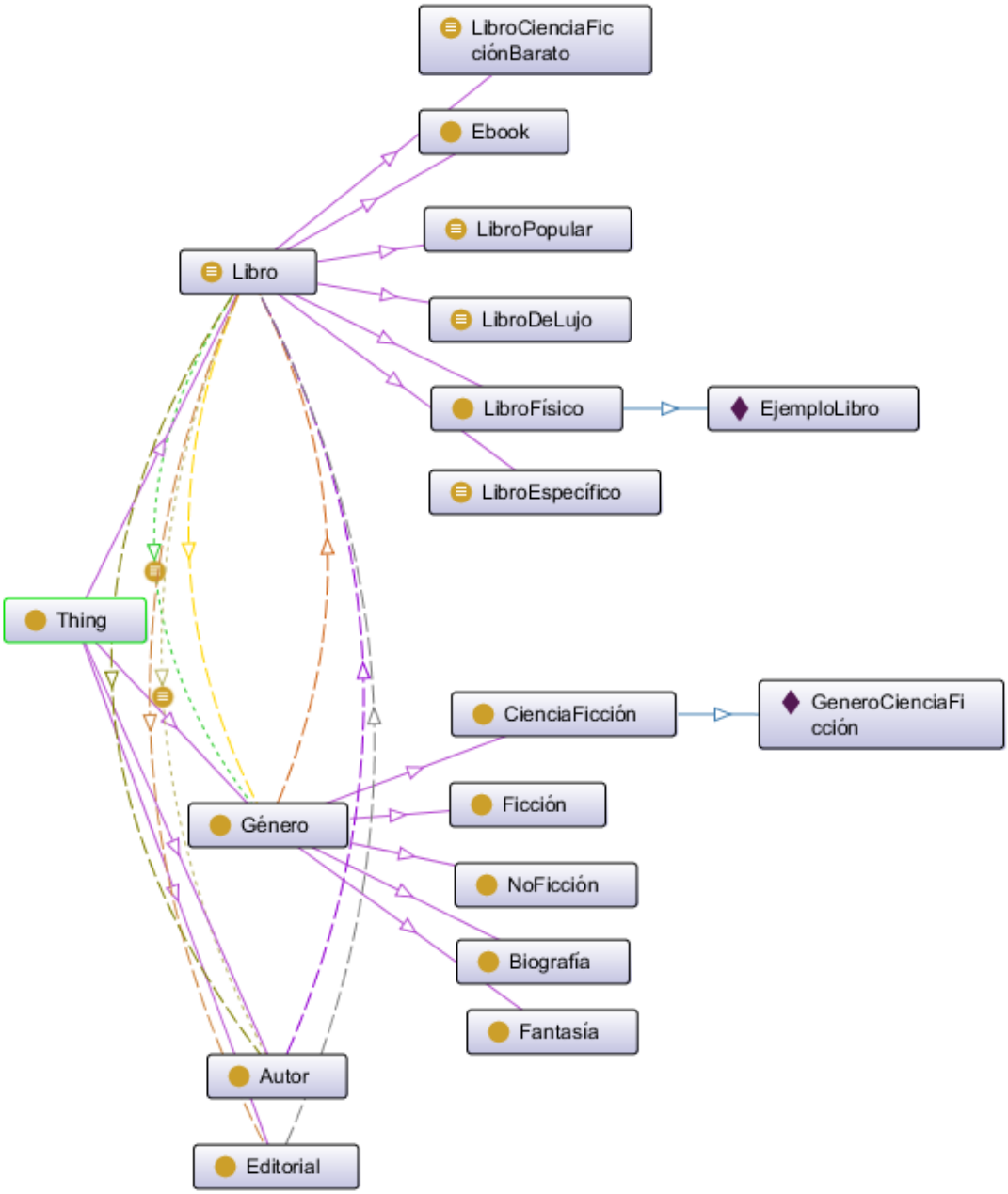

Figura 8: OntoGraf

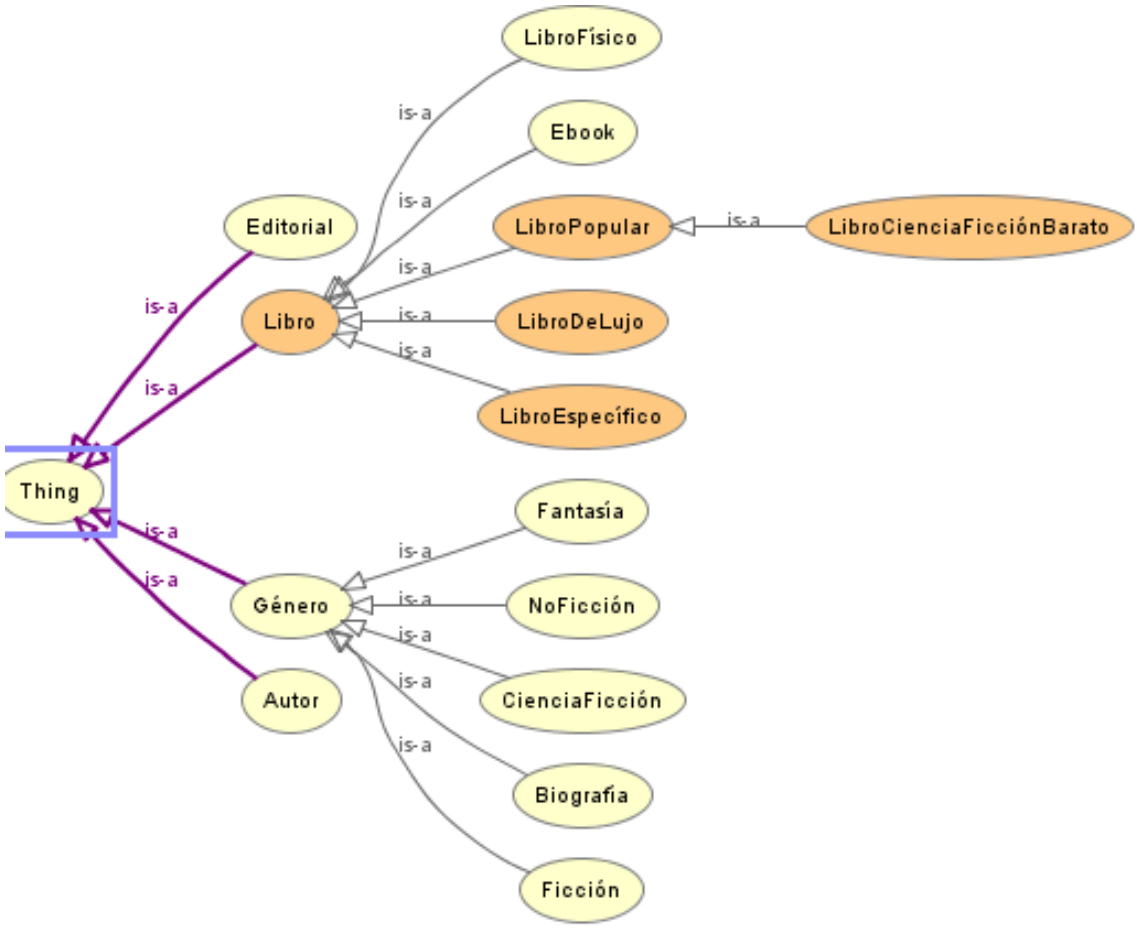

Figura 9: OWLVizCorrecto