

# 本体构建 Protege 基础教程

## 写在前面的话

Ontology，即本体，来源于哲学领域，但自从被图书情报领域专家运用于图书情报领域，便在此领域得到大家的一致认可，各种基于本体的研究论文也层出不穷，但 Protege4.0 以上版本较之 Protege3.X 版本，界面功能发生了很大变化，以前其他学者出的学习教程已经并不适合初入本体领域的学者，而 Protege 官方说明又是全英文解释，给初学者更是带来了很大不便，由此，本人这篇本体构建 Protege 基础教程应运而生，衷心希望可以给其他学者学习本体构建工具以及以后进行基于本体构建领域的研究工作带来便利。在此，特别感谢唐门的 GGJJ 在我学习运用 Protege 过程中给了我很多的理论支持，使我在这个学习过程中思维更加清晰。

——soonfy

学习软件，首先还是看软件版本，本人演示的是 Protege4.1 版本，与 Protege4.0 版本以上的版本界面都较为相似，版本是 4.0 以上的学者，都可以借鉴。另外，本文档主要是界面介绍及逻辑推理，至于本体构建中的个体关联、实体查询请关注下期文档。

## 一、界面介绍

1、打开 Protege 软件。如图 1 所示。

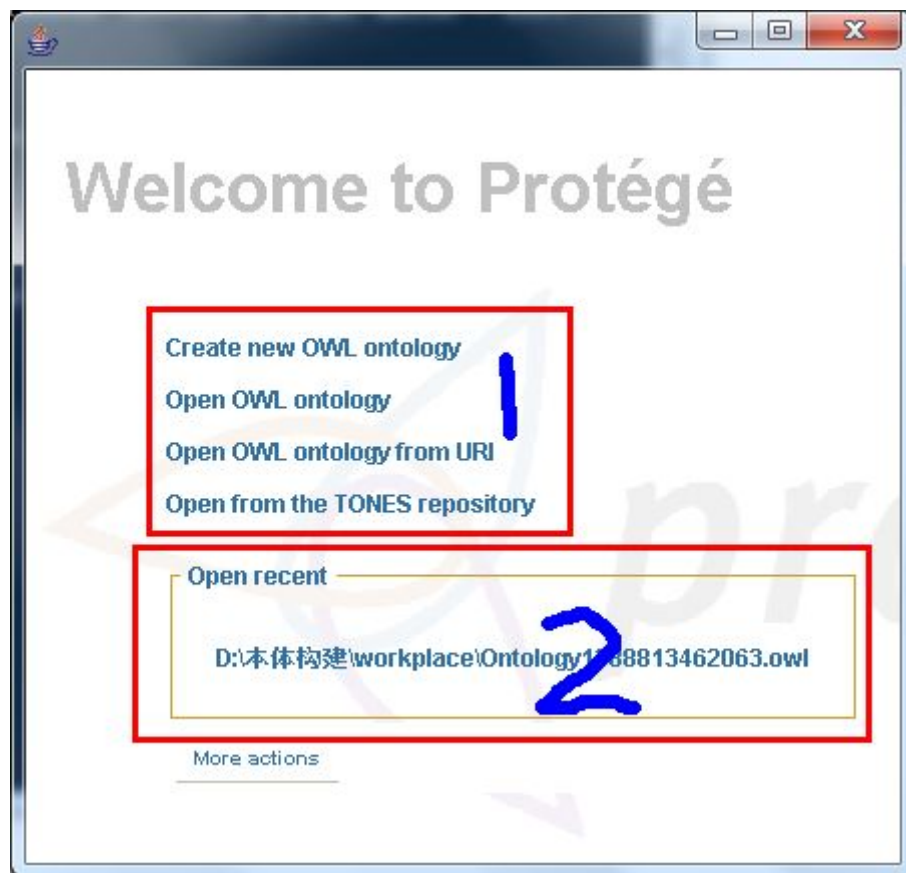


图 1

在图 1 中，方框 1

Create new OWL ontology: 新建 OWL 本体;

Open OWL ontology: 打开一个 OWL 本体;

Open OWL ontology from URI: 通过通用资源标识符 (URI) 打开一个 OWL 本体;

Open from the TONES repository: 从 TONES 库打开 OWL 本体。

方框 2

Open recent: 最近打开的 OWL 本体路径。

More actions: 更多功能。功能有“重新回到默认设置”、“检查更新”。

## 2、新建 OWL 本体文件介绍

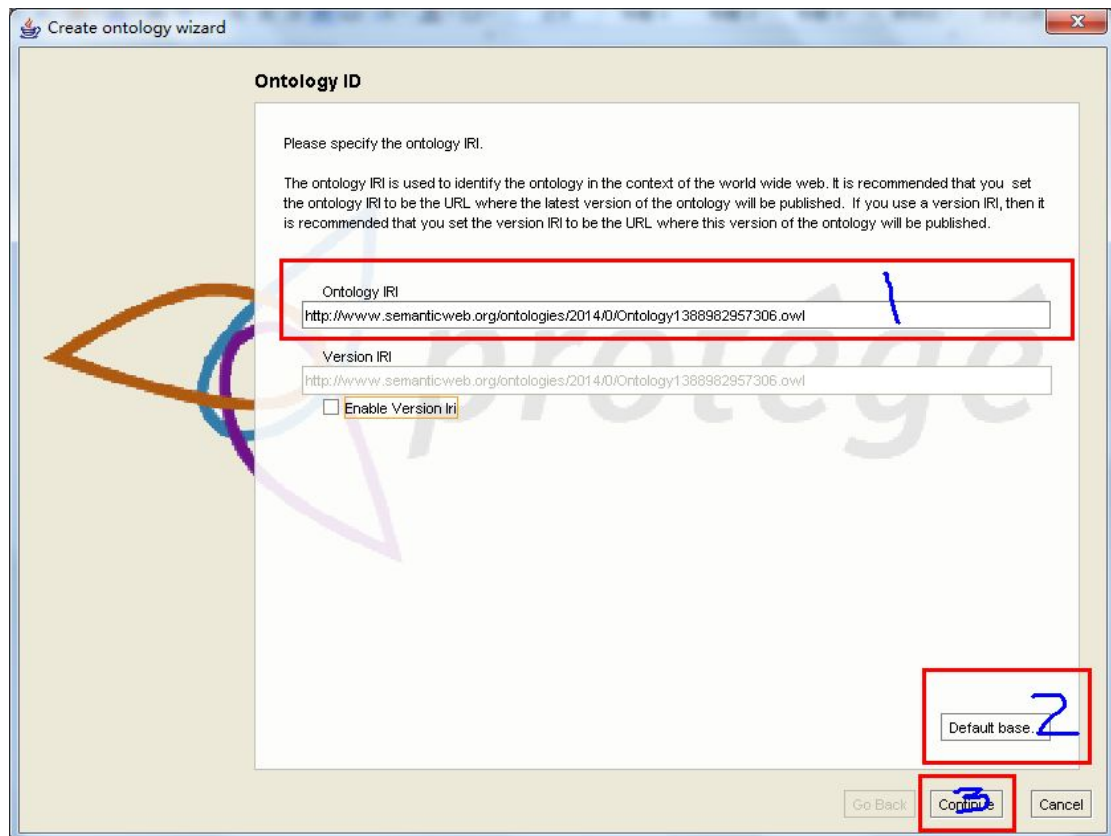


图 2

在图 2 中，方框 1

Ontology IRI: 默认的 IRI 路径（不可随意更改，必须符合 RDF 文件规则）。

方框 2

Default base: 默认 URI 路径信息。

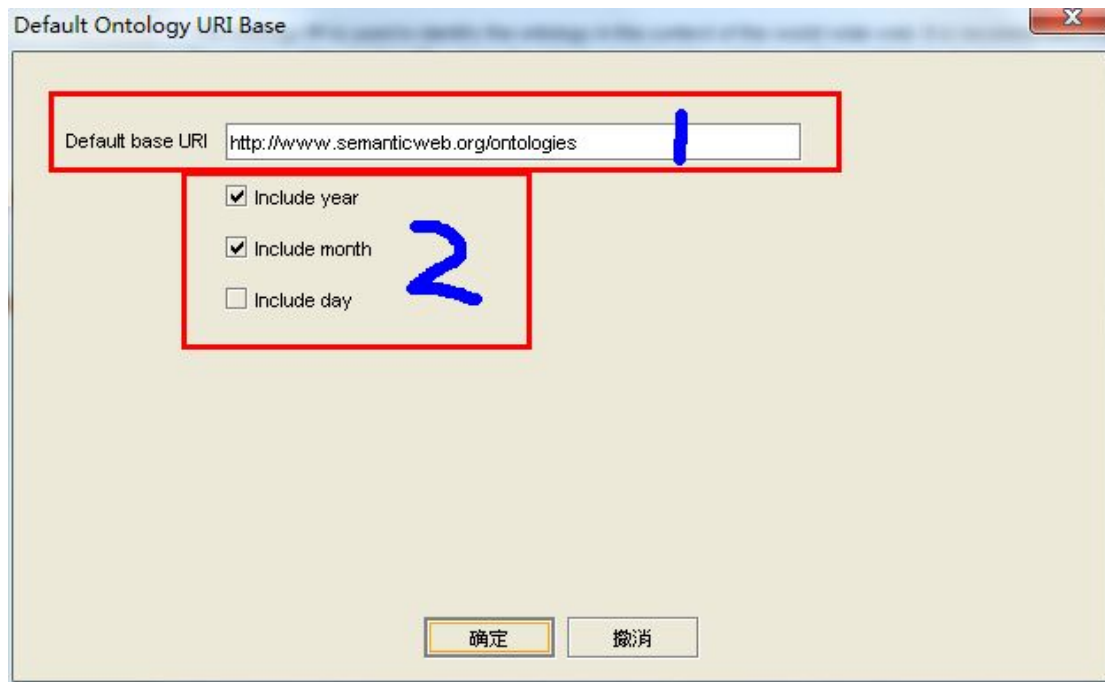


图 3

图 3 为 URI 默认设置，方框 1

Default base URI：默认的 URI 路径。

方框 2

设置默认的 URI 路径中保存哪些信息，年、月、日。

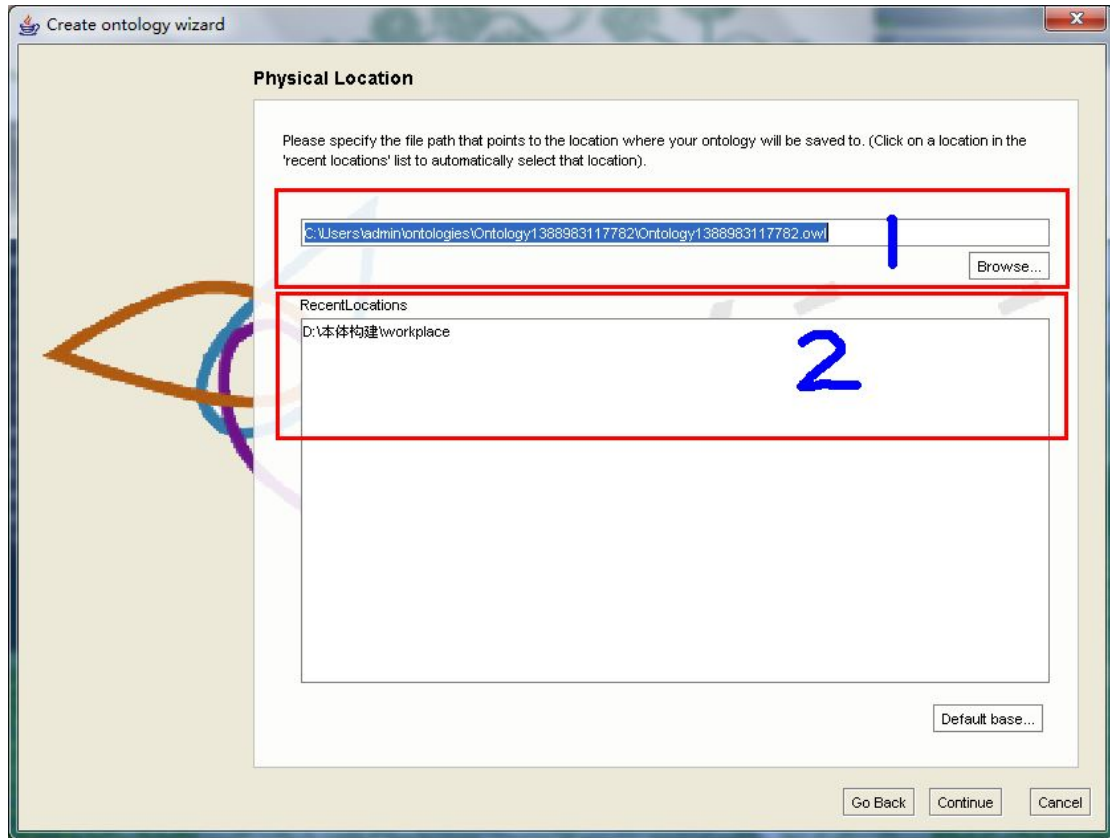


图 4

在图 4 中，方框 1

Physical Location: OWL 物理保存路径。

方框 2

RecentLocations: 最近一次打开 OWL 位置。

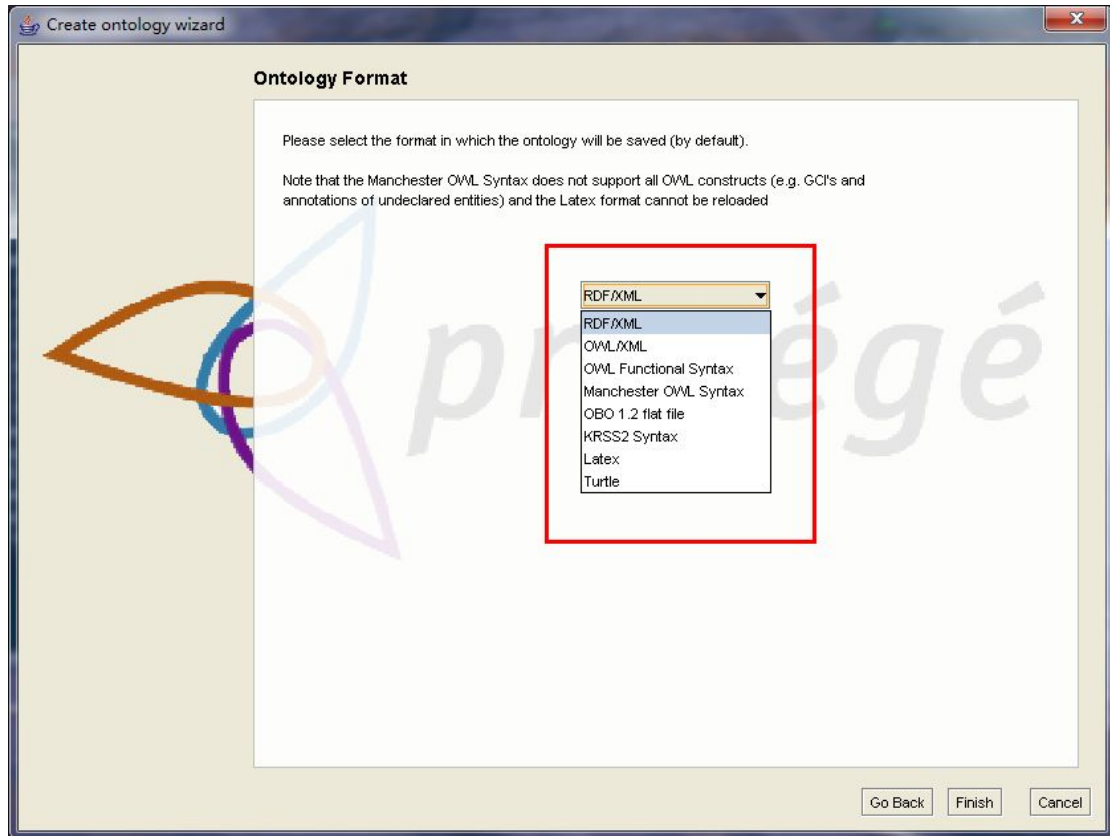


图 5

Ontology Format: 本体标准格式。RDF/XML; OWL/XML; OWL Functional Syntax; Manchester OWL Syntax; OBO 1.2 flat file; KRSS2 Syntax; Latex; Turtle。

### 3、Protege 窗口标签介绍

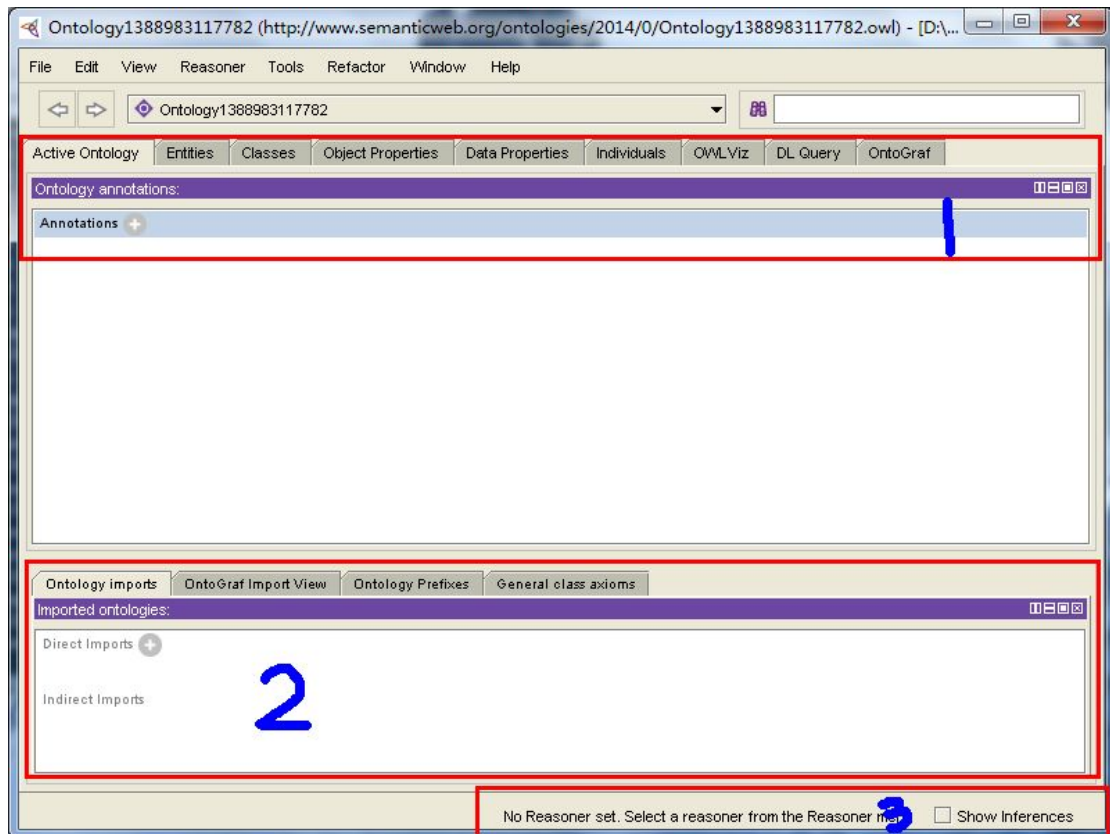


图 6

在图 6 中，Active Ontology 活动本体（大标签）

方框 1

Annotations：注释。

方框 1 中右上角四种调节窗口按钮

Split vertically：水平分离窗口；

Split horizontally：垂直分离窗口；

Float：浮动窗口；

Close：关闭窗口。

方框 2

Ontology imports 导入本体（小标签）

Direct imports：直接导入本体；

Indirect imports: 间接导入本体。

OntoGraf Import View: 本体导入视图（小标签）

Ontology Prefixes: 本体前缀（小标签）

General class axioms: 通用类公理（小标签）

方框 3

状态栏

No Reasoner set: 没有加载推理机。

Show Inferences: 显示推理过程。

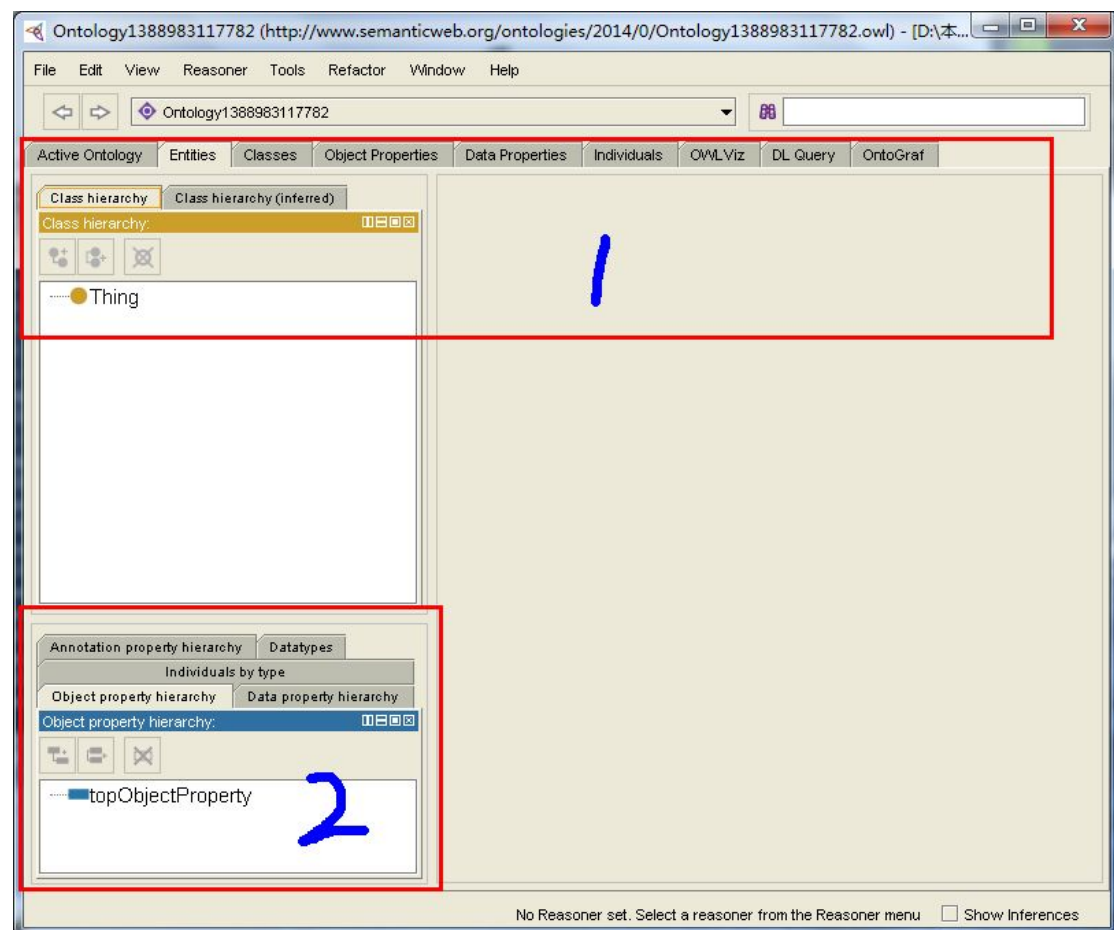


图 7

在图 7 中, Entities: 实体（大标签）

方框 1



Class hierarchy: 类层次结构（小标签）

Class hierarchy(inferred): 推断的类层次结构（小标签）

方框 2

Annotation property hierarchy: 注释关联层次结构（小标签）

Datatypes: 数据类型（小标签）

Data property hierarchy: 数据关联层次结构（小标签）

Individuals by type: 个体类型（小标签）

Object property hierarchy: 事物关联层次结构（小标签）

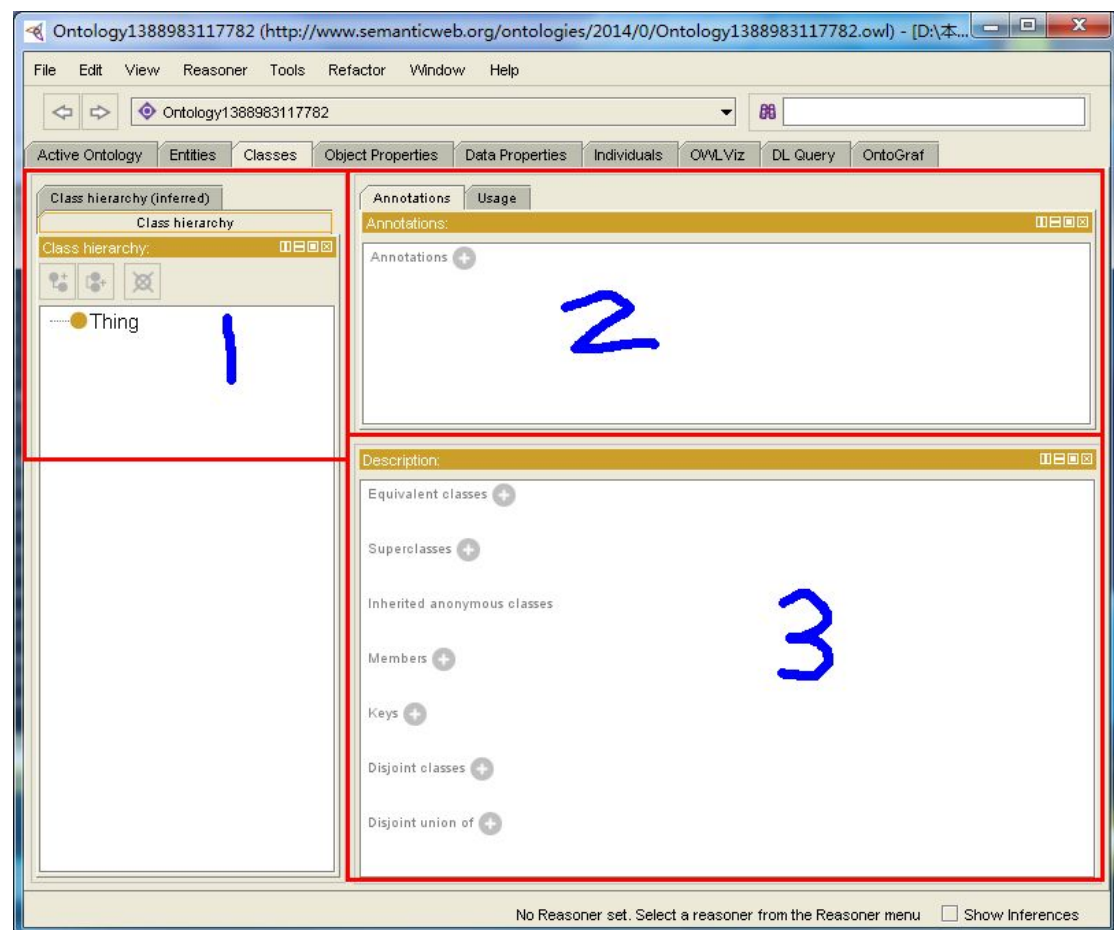





图 8

在图 8 中，Classes: 类（大标签）

方框 1

Class hierarchy: 类层级结构（小标签）

Class hierarchy(inferred): 推理的类层级结构（小标签）

三个增删类功能:   

 subclass 添加子类;  sibling class 添加同级类;  delete 删除类。

方框 2

Annotation: 注释（小标签）

Usage: 类使用情况（小标签）类的一级超类和子类关系

方框 3 Description: 描述（小标签）

Equivalent classes: 等同的类, 用来定义与当前类相等等同的类, 或 not class 不相等的类, 主要用来推理, 相当于充分必要条件

Superclasses: 超类, 用来定义当前类的父类, 限制类。Some 存在量词（可简单译为“可以”）, Only 全称量词（可简单译为“只”）, Min 基数量词最小值, Max 基数量词最大值, Exactly 基数量词准确值。这些是约束属性的条件, 用来限制对象属性的, 相当于必要条件

Inherited anonymous classes: 继承匿名的类, 继承父类的关联关系, 匿名类定义在父类中, 被子类继承, 是子类的必要条件

Members: 类的成员, 类所包含的个体

Keys: 关键字, 描述类的行为和属性

Disjoint classes: 不相交的类, 互斥的类, 不存在一个个体同时属于两个 Disjoint class

Disjoint union of: 不想交类集合, 互斥的类的集合, 一个类的所有子类都不相交

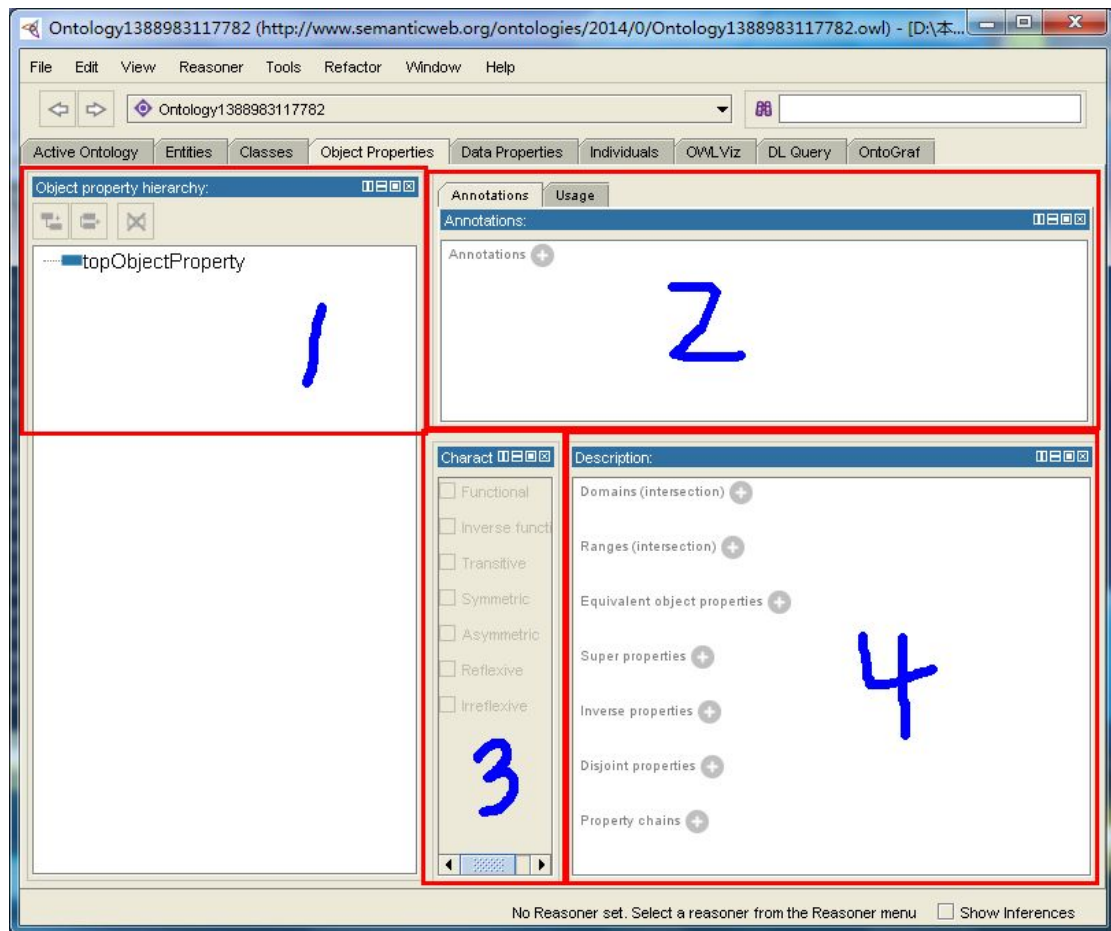


图 9

在图 9 中, Object Properties: 事物关联 (大标签)

方框 1 Object property hierarchy: 事物关联层级结构 (小标签)

三个增删关联功能

sub property 添加子关联; sibling property 添加同级关联;

delete 删除关联。

方框 2

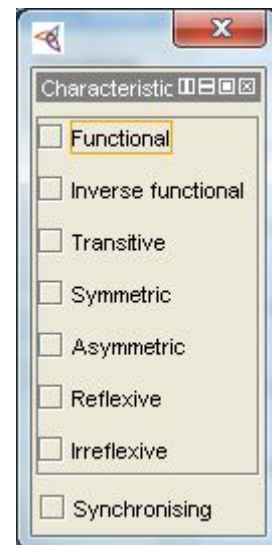
Annotation: 注释 (小标签)

Usage: 事物关联使用情况（小标签）事物关联的一级父关联和子关联

方框 3 Characteristics: 特性（小标签）

Functional: 函数性，标注了这个特性，说明该对象属性只能连接一个个体，假如连接的两个人，说明两个个体是同一个个体。例如小明最好的朋友是小强，小明最好的朋友是小光，则小光就是小强。

Inverse functional: 逆函数性，是 Functional 函数性的反性质，对于一个给定的个体，只有最多一个个体能通过该属性连接那个个体



Transitive: 传递性， $a > b$ ,  $b > c$  推出  $a > c$

Symmetric: 对称性，两个个体对称，具有对称性，不能具有函数性和逆函数性。例如小明和小光是朋友，小光和小明也是朋友。

Asymmetric: 不对称性

Reflexive: 自反性，自己和自己相关联。例如小明喜欢自己。

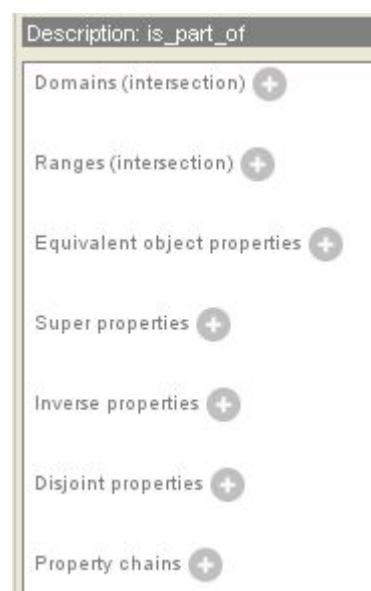
Irreflexive: 不自反性

方框 4 Description: 描述（小标签）

Domains(intersection): 定义域

Ranges (intersection): 值域

关联连接的是定义域的个体到值域的个体，对象属性的定义域和值域主要用



来推理。例如小明吃饭，“小明”就是定义域，“饭”就是值域，“吃”是关联。

Equivalent object properties: 同等事物关联

Super properties: 父关联

Inverse properties: 逆关联，就是相反关联。例如小明是小光的父亲，小光是小明的儿子，“父亲”和“儿子”就是相反关联。

Disjoint properties: 互斥关联

Property chain: 关联链

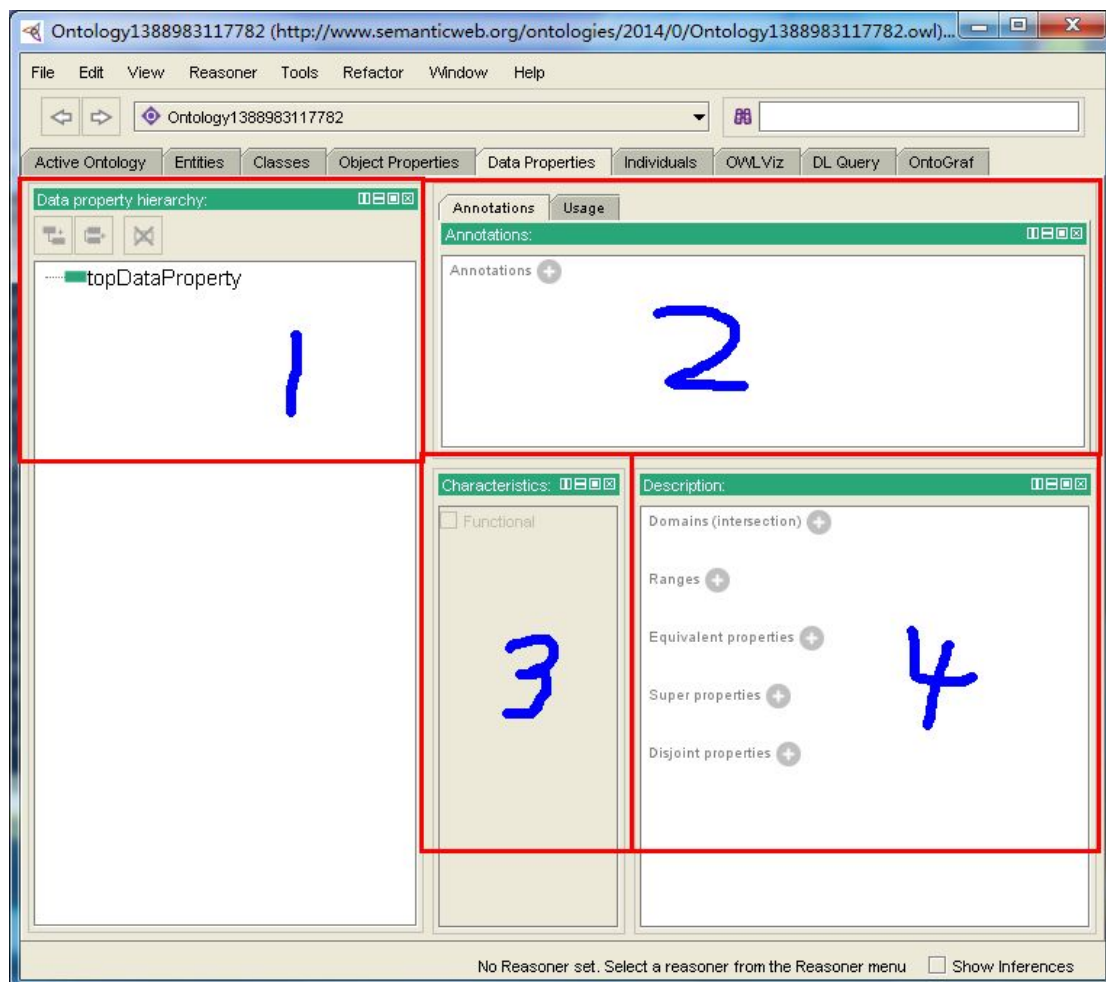








图 10

在图 10 中，Data Properties: 数据关联（大标签）

方框 1 Data property hierarchy: 数据关联层级结构（小标签）

三个增删数据关联的功能   

 sub property 添加子关联；  sibling property 添加同级关联；  
 delete 删除关联

方框 2

Annotation: 注释（小标签）

Usage: 数据关联使用情况（小标签）[数据关联的一级父关联和子关联](#)

方框 3 Characteristics: 特性（小标签）

[用来连接个体和 XML Schema 数据类型值或 rdf literal](#)，该属性不能为传递的，对称的，反函数的，只可以为函数的。

方框 4 Description: 描述（小标签）

Domains(intersection): 定义域

Ranges (intersection): 值域

Equivalent object properties: 同等数据关联

Super properties: 父数据关联

Disjoint properties: 互斥数据关联

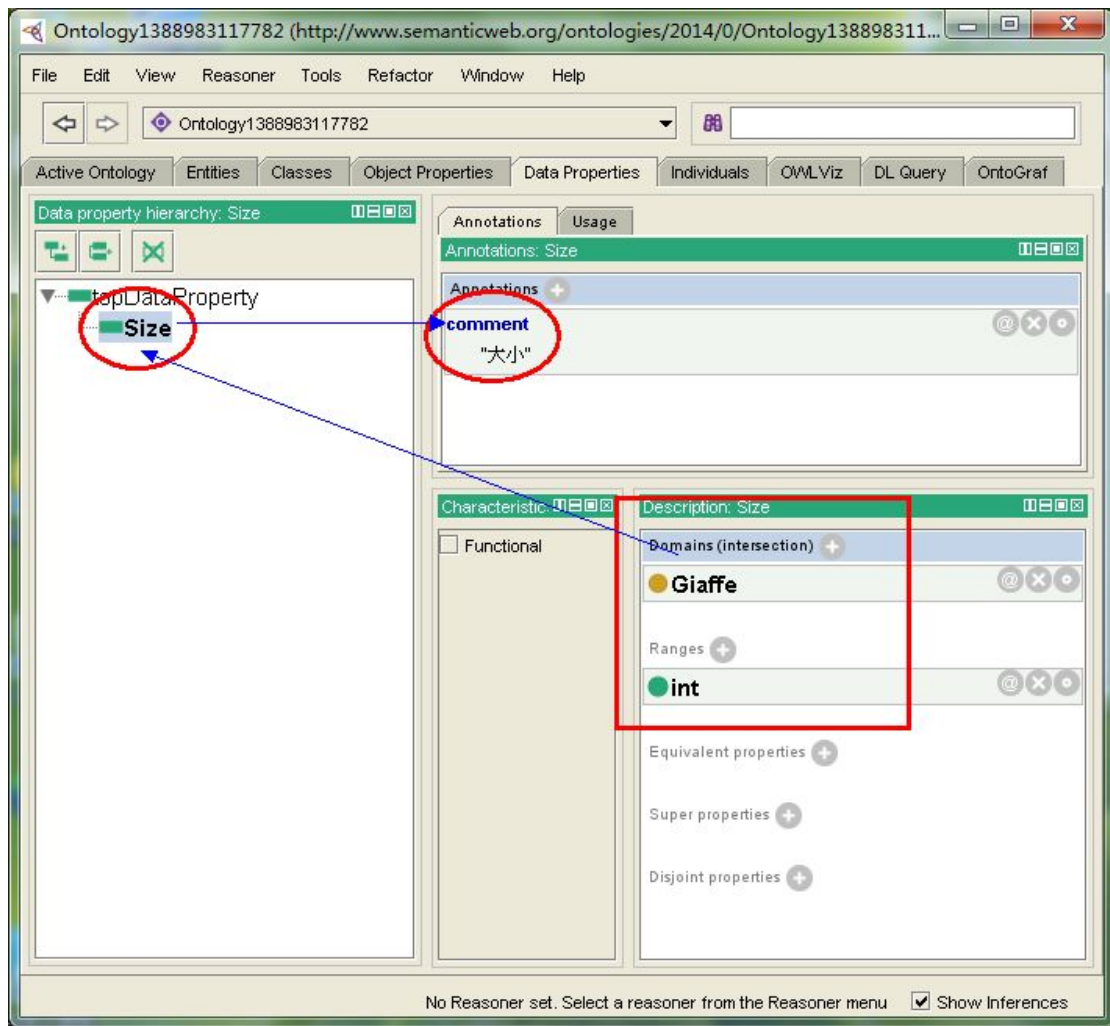


图 11

在图 11 中，描述的是类“Giaffe”的个体具有“Size”数据属性，其值得类型为“int”；数据属性“Size”的 Annotation comment 为“大小”。



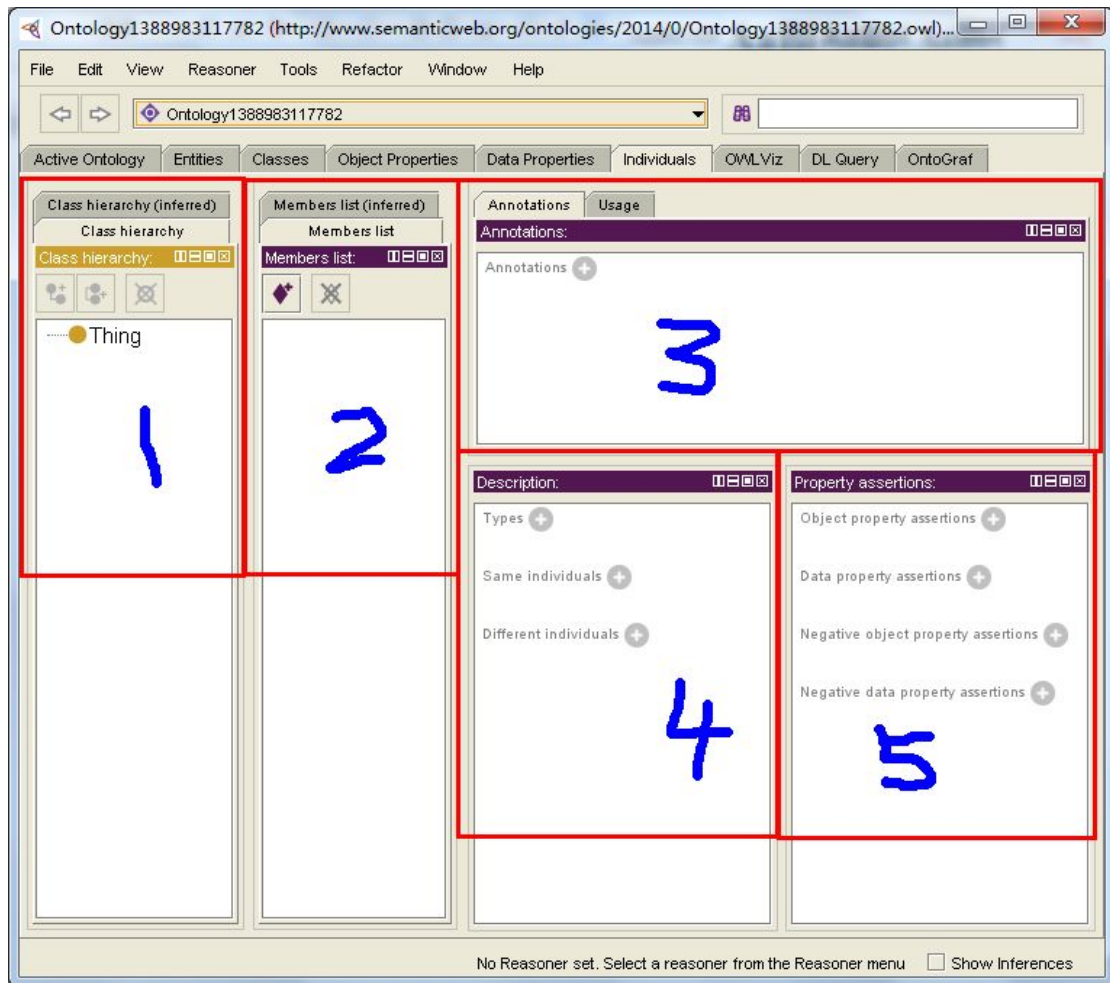


图 12

在图 12 中，Individual：个体（大标签）

方框 1

Class hierarchy：类层级结构（小标签）

Class hierarchy(inferred)：推理的类层级结构（小标签）

方框 2

Members list：成员列表（小标签）

Members list(inferred)：推理的成员列表

两个增删成员功能  

 add 添加成员；  delete 删除成员



### 方框 3

Annotation: 注释 (小标签)

Usage: 成员使用情况 (小标签) [成员的一级父类和成员间数据关联](#)

方框 4 Description: 描述 (小标签)

Types: 类型, [这个成员所属的父类](#)

Same individual: 相同的个体

Different individual: 不同的个体

方框 5 Property assertions: (小标签)

Object property assertions: 对象关联声明, [通过事物关联其它个体](#)

Data property assertions: 数据关联声明, [设置个体数据关联](#)

Negative object property assertions: 否定对象关联声明

Negative data property assertions: 否定数据关联声明



Asserted model 声明模型，即使定义模型，Inferred model 推理后的模型，即定义模型经过推理机进行推理后的模型。

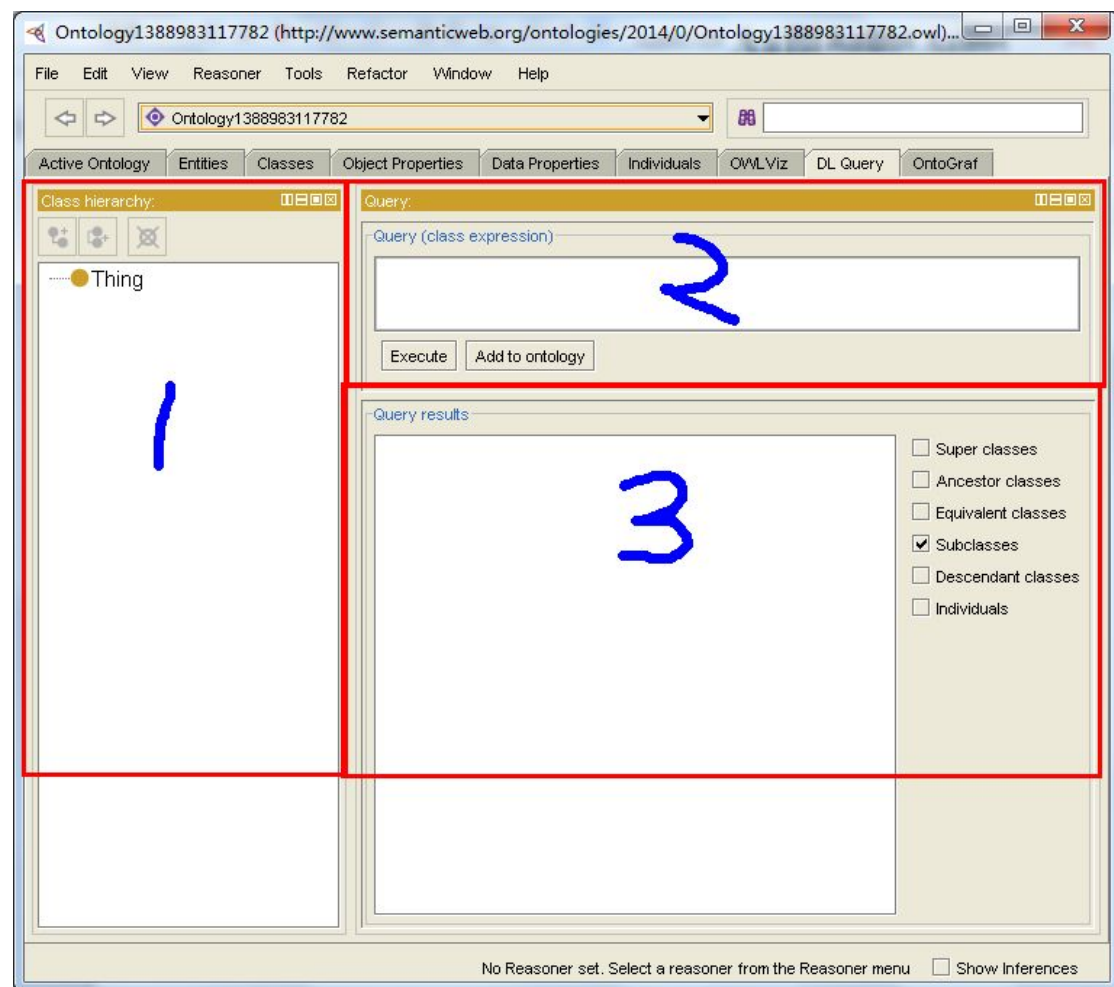


图 14

在图 14 中，DL Query: DL 查询（大标签）

方框 1 Class hierarchy: 类层级结构（小标签）

方框 2 Query: 查询，输入需要进行推理的查询语句

Execute: 查询执行

Add to ontology: 添加到本体

方框 3 Query results: 查询结果

Super classes: 查询结果显示超类

Ancestor classes: 查询结果显示祖先类

Equivalent classes: 查询结果显示等同的类

Subclasses: 查询结果显示子类

Descentdant classes: 查询结果显示子孙类

Individual: 查询结果显示个体

OntoGraf: 本体图形界面（大标签）

方框 1 Class hierarchy: 类层级结构（小标签）

方框 2 OntoGraf: 本体图形（小标签）



图 15

在图 15 中，Search: 查找符合某种条件的类的本体图形

Clear: 清除本体图形



主要功能依次是显示本体，移除孤立节点，grid alphabetical 网格方式显示本体，Radial 射线方式显示本体，Spring 生长图方式显示本体，竖向树方式显示本体，横向树方式显示本体，vertical directed 竖向显示本体，横向显示本体，图形放大，图形无放缩，图形缩小，节点显示类型，连线显示类型，转换为图片输出，设置节点提示信息，保存图形，打开图形，以 DOT 格式输出图形，显示节点提示信息。

#### 4、Protege 窗口菜单介绍

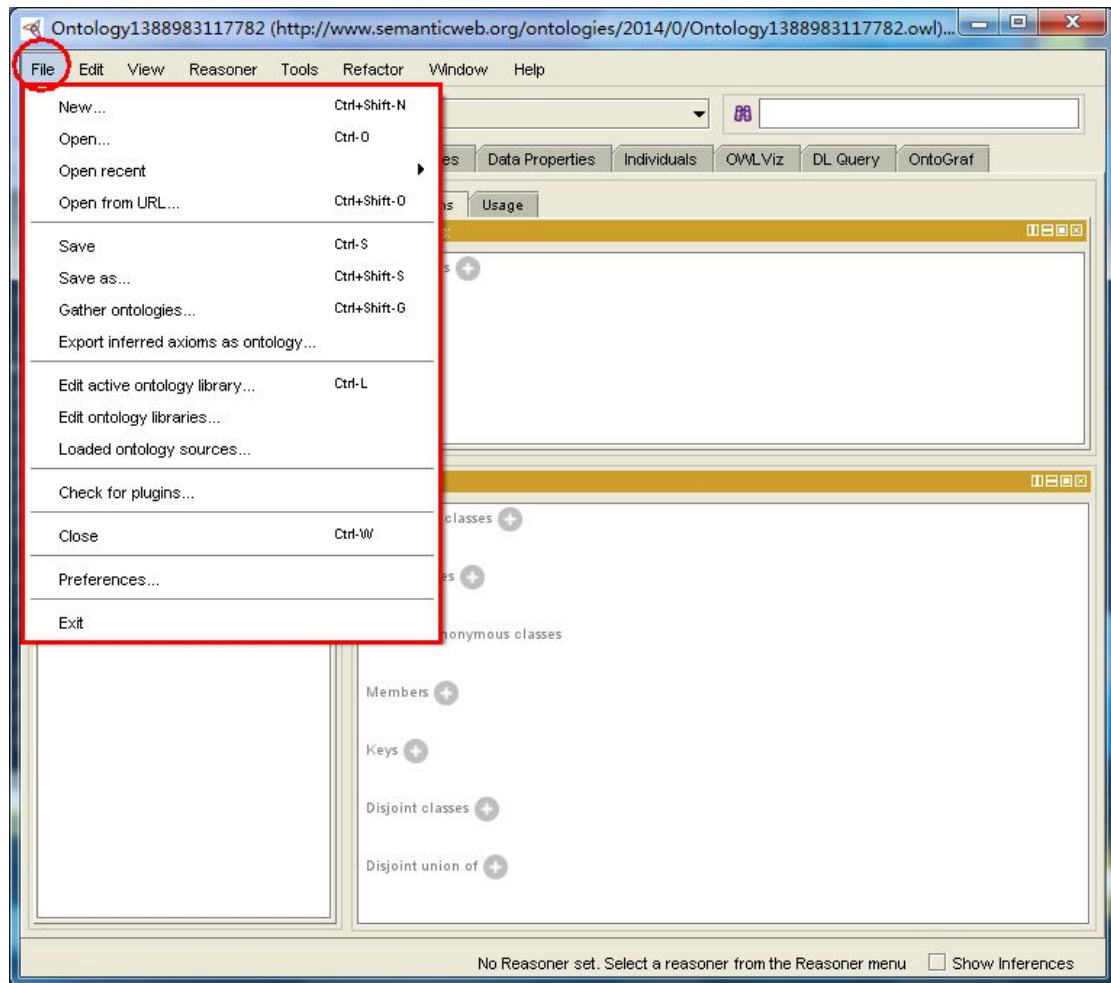


图 16

在图 16 中，File（文件）菜单

New：新构建本体文件；

Open：打开构建本体文件；

Open recent：打开最近打开的本体文件；

Open from URL：通过 URL 格式打开本体文件；

Save：保存本体文件；

Save as：另存为本体文件；

Gather ontologies: 另存多个本体文件; 必须是在同一个本体窗口打开的多个本体;

Export inferred axioms as ontology: 将推理后的本体另存为一个本体文件, 首先推理机需要启动, 其次, 如果 FaCT++推理机导出本体可能出错, 就换 HermiT 推理机推理然后再导出;

Edit active ontology library: 编辑活动本体库, 打开的是本体格式文件;

Edit ontology libraries: 编辑本体库, 打开的是 XML 格式文件;

Loaded ontology sources: 显示本体文件的物理和 URI 来源;

Check for plugins: 检查更新;

Close: 关闭当前本体构建窗口;

Preferences: 选项设置;

Exit: 退出 protege 软件。

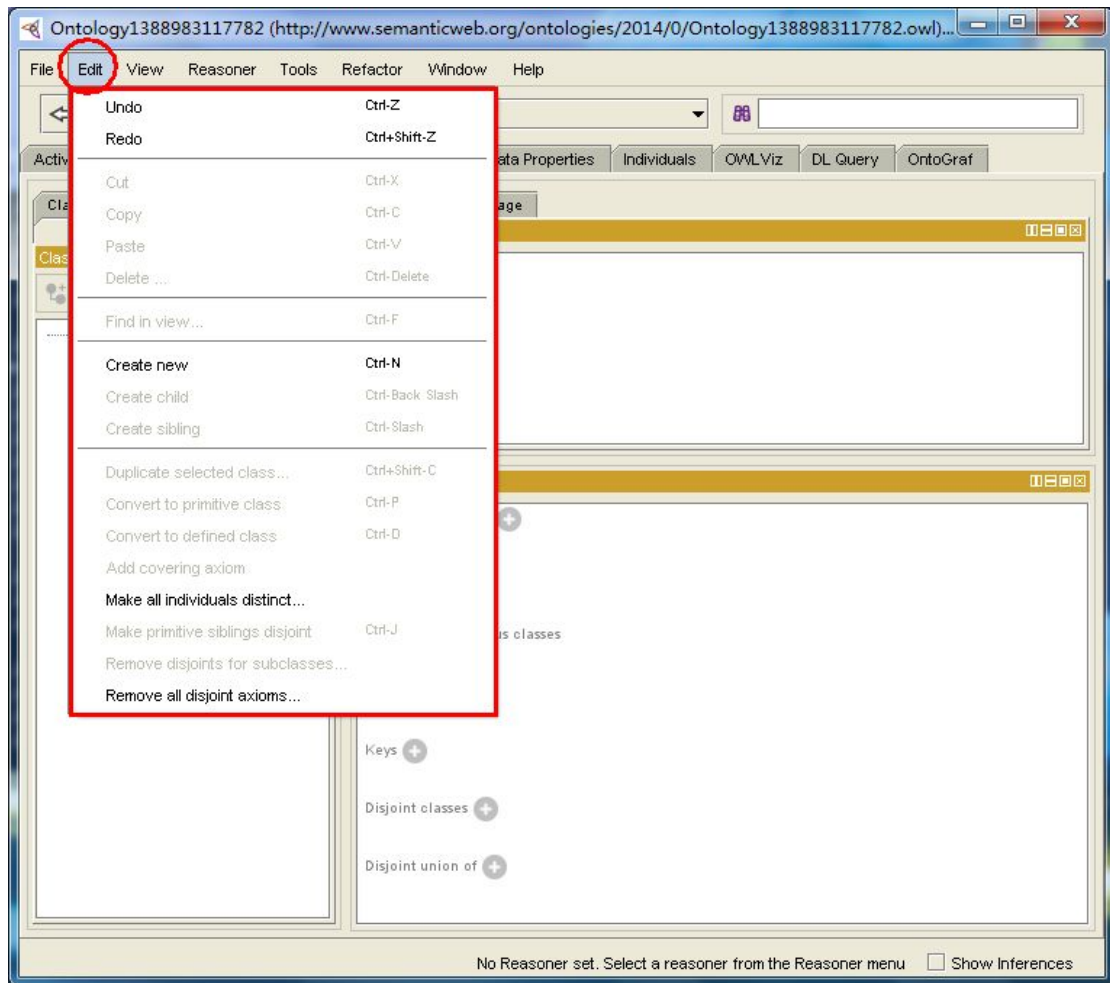


图 17

在图 17 中，Edit（编辑）菜单

Undo：撤消；

Redo：重做；

Cut：剪切；

Copy：复制；

Paste：粘贴；

Delete：删除；

Find in view：查找，界面显示为灰色不可用状态，但可以使用快捷键 **Ctrl+F** 查找；

Create new: 新建类, 只能创建 Thing 子类;

Create child: 新建当前选定类的子类;

Create sibling: 新建当前选定类的同级类;

Duplicate selected class: 复制当前类, 与 Copy 区别在于, 它也同时复制子类, 等价类, 注释等其它内容;

Convert to primitive class: 把等价类描述内容全部剪切到父类描述里面;

Convert to defined class: 把父类描述内容全部剪切到等价类描述里面;

Add covering axiom: 添加隐含公理, 把选定类的所有子类添加到父类描述里面;

Make all individuals distinct: 设定所有的个体互不相同;

Make primitive siblings disjoint: 设置选定类的同级类互不相交;

Remove disjoint for subclasses: 移除当前选定类的子类多有不交交的公理;

Remove all disjoint axioms: 移除所有不想交的公理;



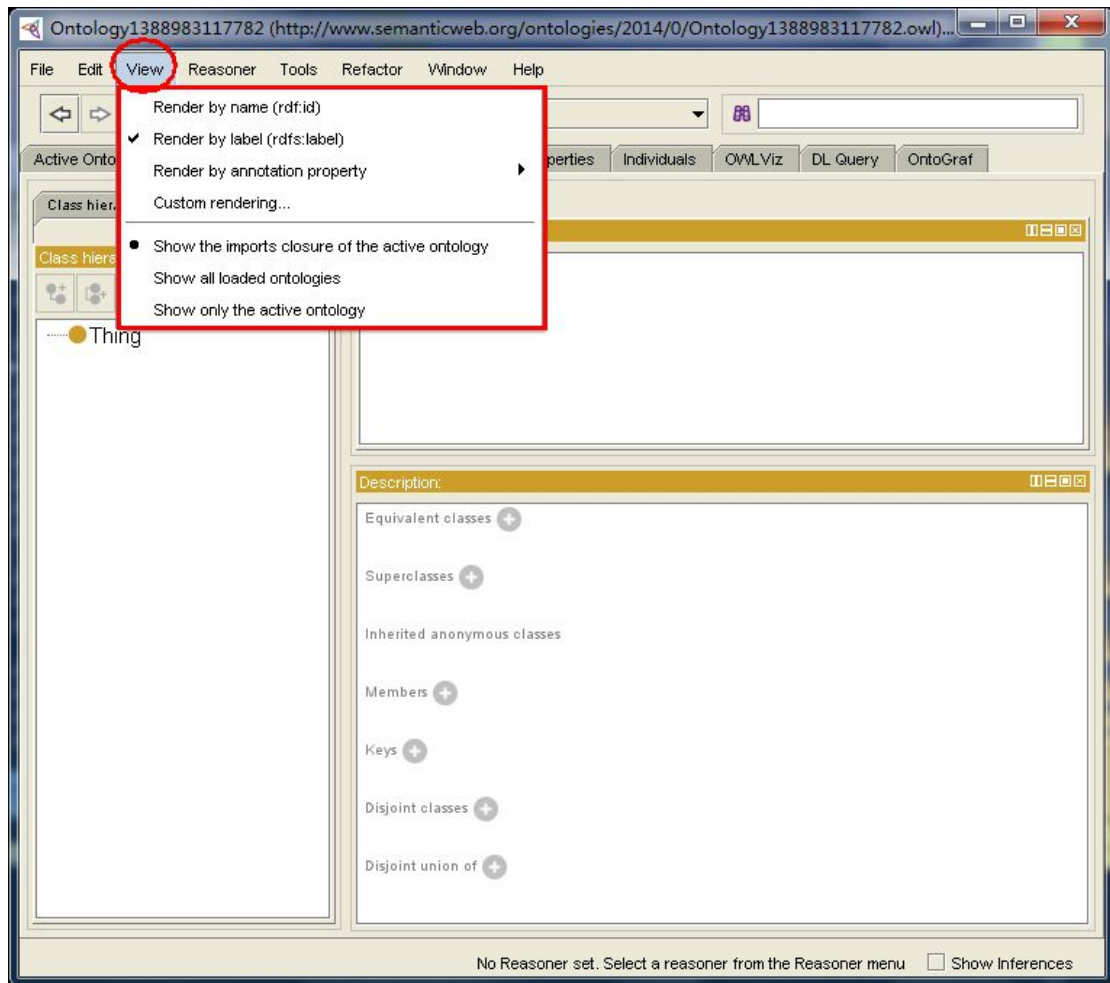


图 18

在图 18 中，view（查看）菜单

Render by name: 按名字显示;

Render by label: 按标签显示;

Render by annotation property: 按描述内容显示;

Custom rendering: 用户自定义显示;

Show the imports closure of the active ontology: 显示活动本体的导入完成;

Show all loaded ontologies: 显示所有已加载的本体;

Show only the active ontology: 只显示活动的本体。

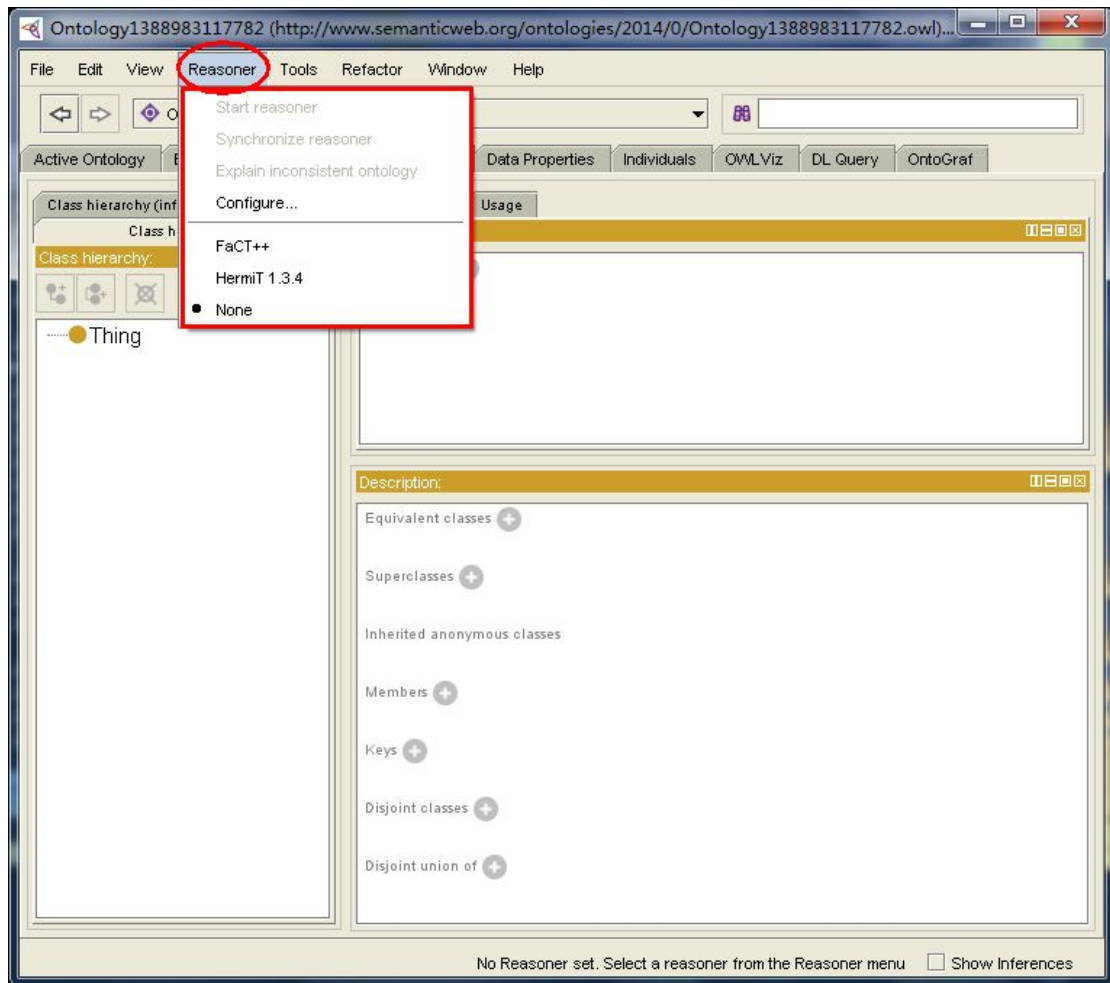


图 19

在图 19 中，Start reasoner: 启动推理机；

Synchronize reasoner: 并发推理机；

Explain inconsistent ontology: 解释不一致的本体；

Configure: 配置属性；

FaCT++: 推理机 FaCT++；

HermiT 1.3.4: 推理机 HermiT；

None: 关闭推理机。

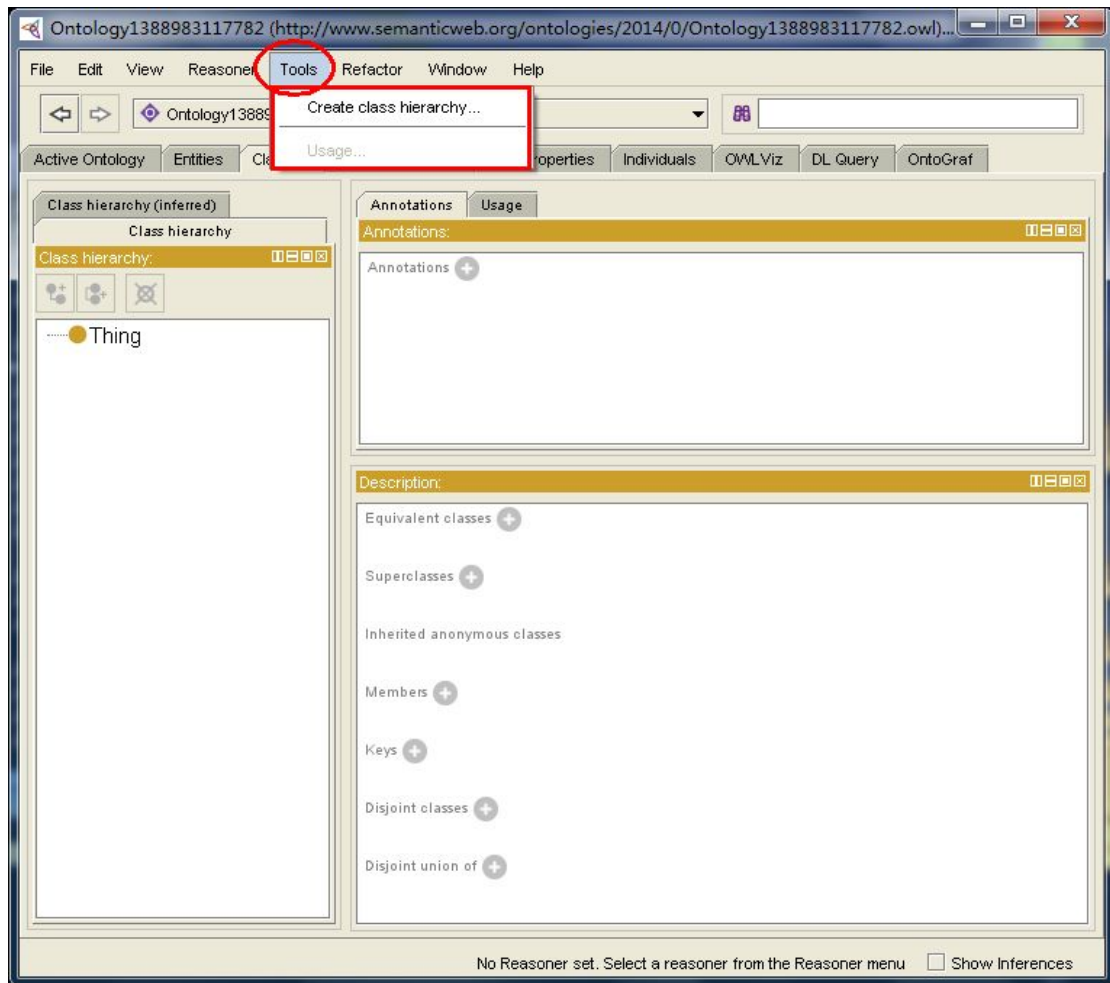


图 20

在图 20 中，Tools（工具）菜单

Create class hierarchy：建立类层级结构；

Usage：打开当前选定项的使用窗口；

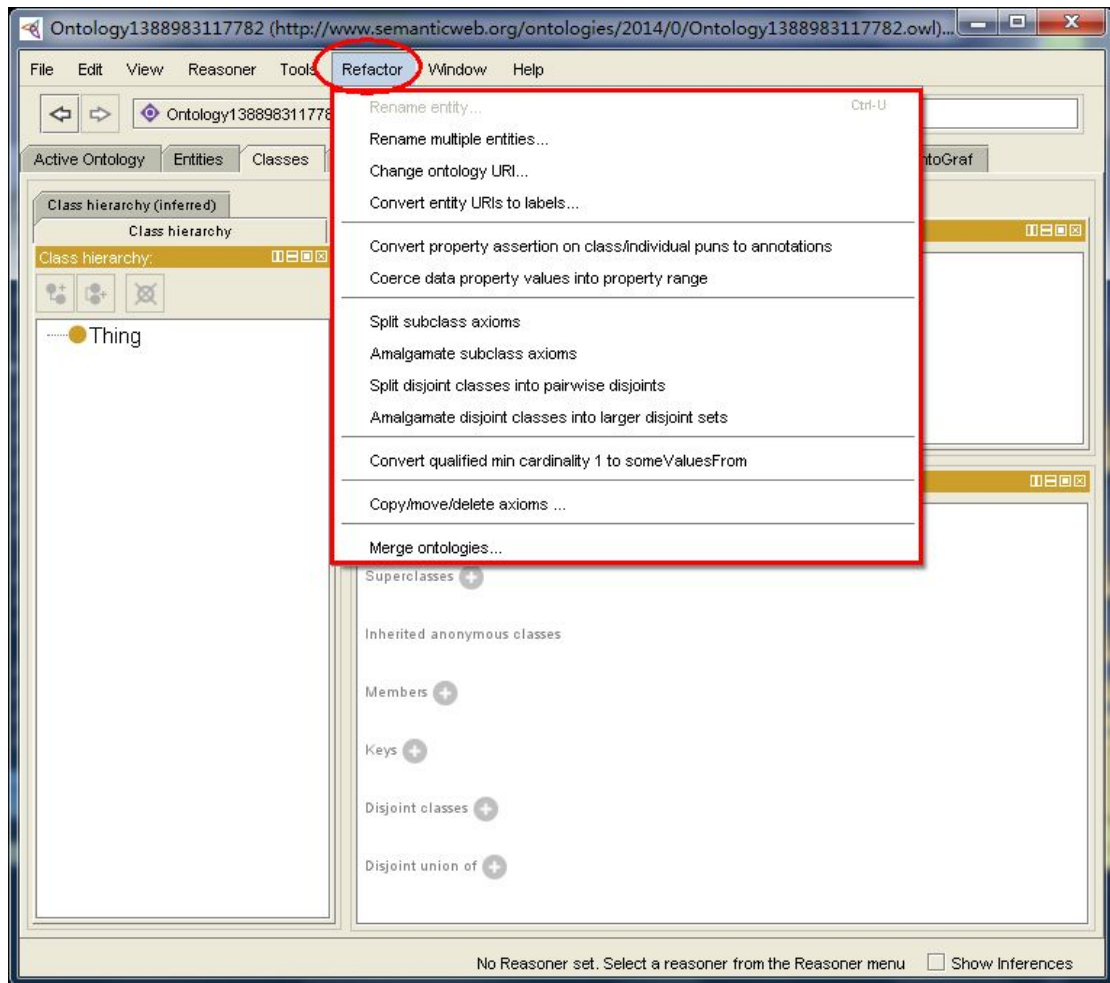


图 21

在图 21 中，Refactor（重构）菜单

Rename entity: 重命名一个实体；

Rename multiple entities: 重命名多个实体；

Change ontology URI: 修改本体 URI；

Convert entity URIs to labels: 转变实体 URIs 为标签；

Convert property assertion on class/individual puns to annotation: 把关联声明转换为类或者个体的注释；

Coerce data property values into property range: 强制数据关联值属于关联值域；

Split subclass axioms: 分裂子类公理, 如果 a 是 (b 和 c) 的子类, 就可以分裂为 a 既是 b 的子类, 又是 c 的子类;

Amalgamate subclass axioms: 合并子类公理, 与上个重构过程相逆;

Split disjoint classes into pairwise disjoint: 分裂不相交类集为一个个成对的不相交类;

Amalgamate disjoint classes into large disjoint sets: 合并不想交的类为一个不想交的类集, 与上个重构过程相逆;

Convert qualified min cardinality 1 to someValuesForm: 把最小限制为 1 的基数转变为 some 形式;

Copy/move/delete/ axioms: 复制、移动、删除公理;

Merge ontologies: 合并本体。

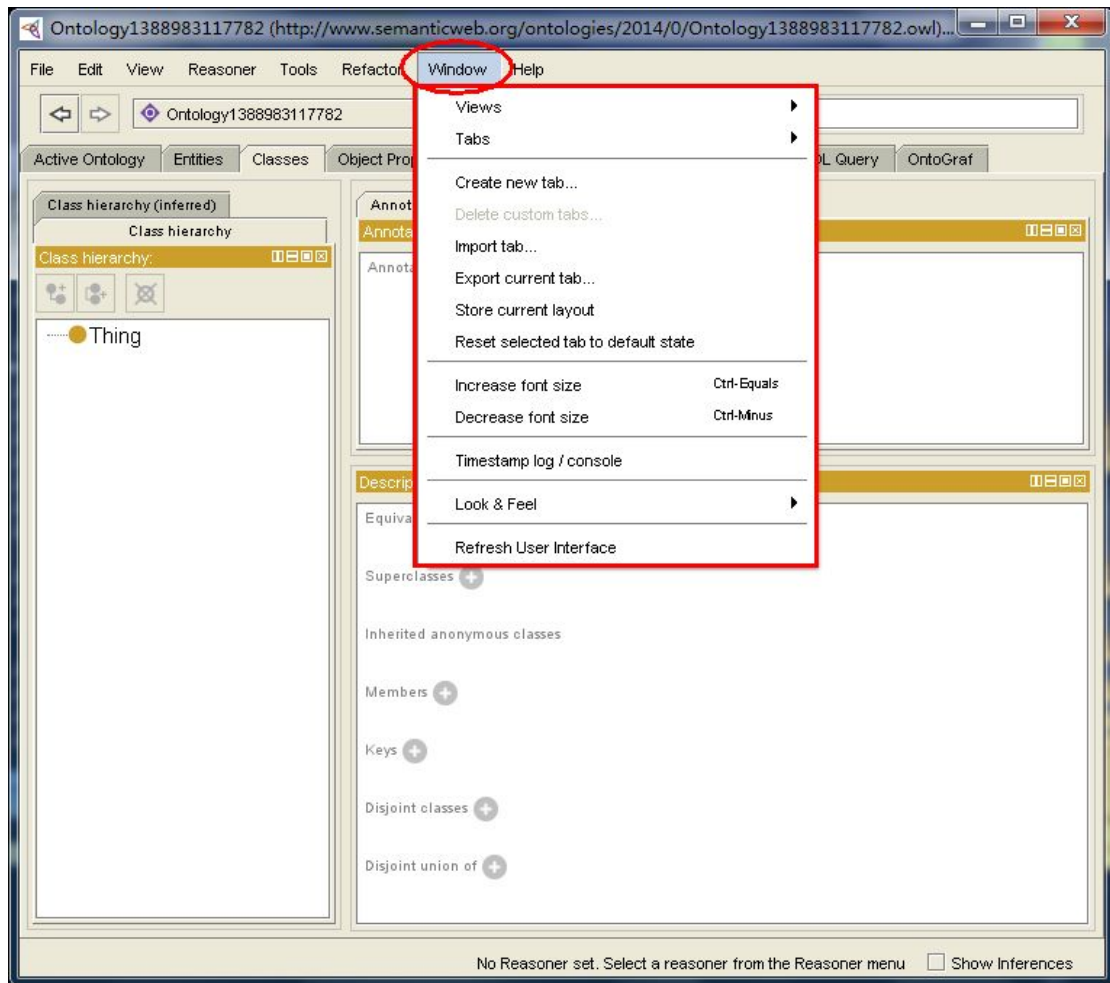


图 22

在图 22 中，Window（窗口）菜单

View: 添加小窗体;

Tabs: 增删标签;

Creat new tab: 新建一个标签;

Delete custom tab: 删除用户新建的标签;

Import tab: 导入标签;

Export current tab: 导出当前标签;

Store current layout: 存储当前布局;

Reset selected tab to default state: 重置标签默认设置;

Increase font size: 增大字体;

Decrease font size: 减小字体;

Timestamp log/console: 添加时间戳日志;

Look & Feel: 窗口样式;

Refresh user interface: 刷新用户窗口。

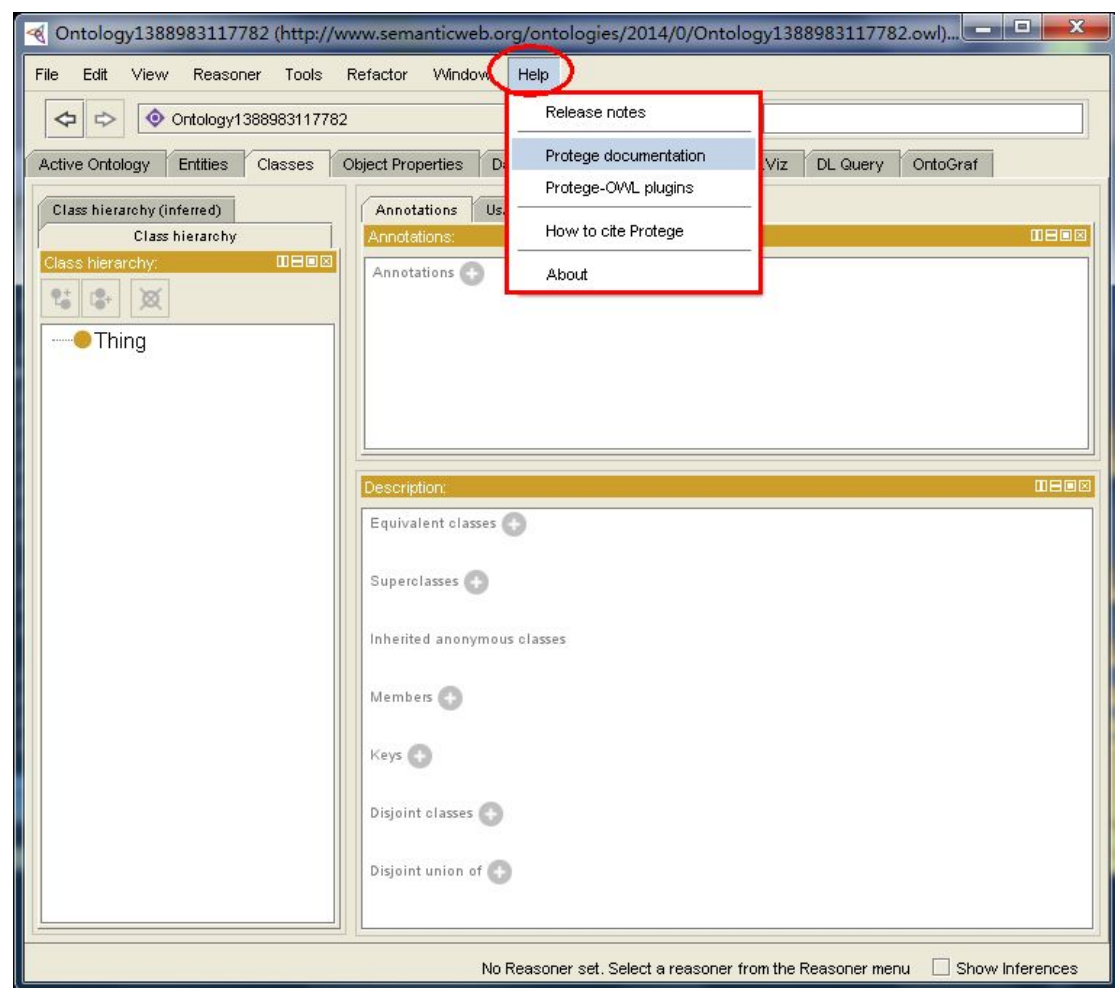


图 23

在图 23 中，Help（帮助）菜单

Release notes: 发布的说明文档;

Protege documentation: Protege 文档;

Protege-OWL plugins: Protege 本体插件;

How to cite protege: 如何引用 protege;

About: 关于。

## 二、推理实验过程

1、打开 Protege 软件，不能随意修改本体 IRI 路径（必须符合 RDF 文件规则），可以根据用户意愿修改物理保存路径。打开的窗口如图 24 所示。

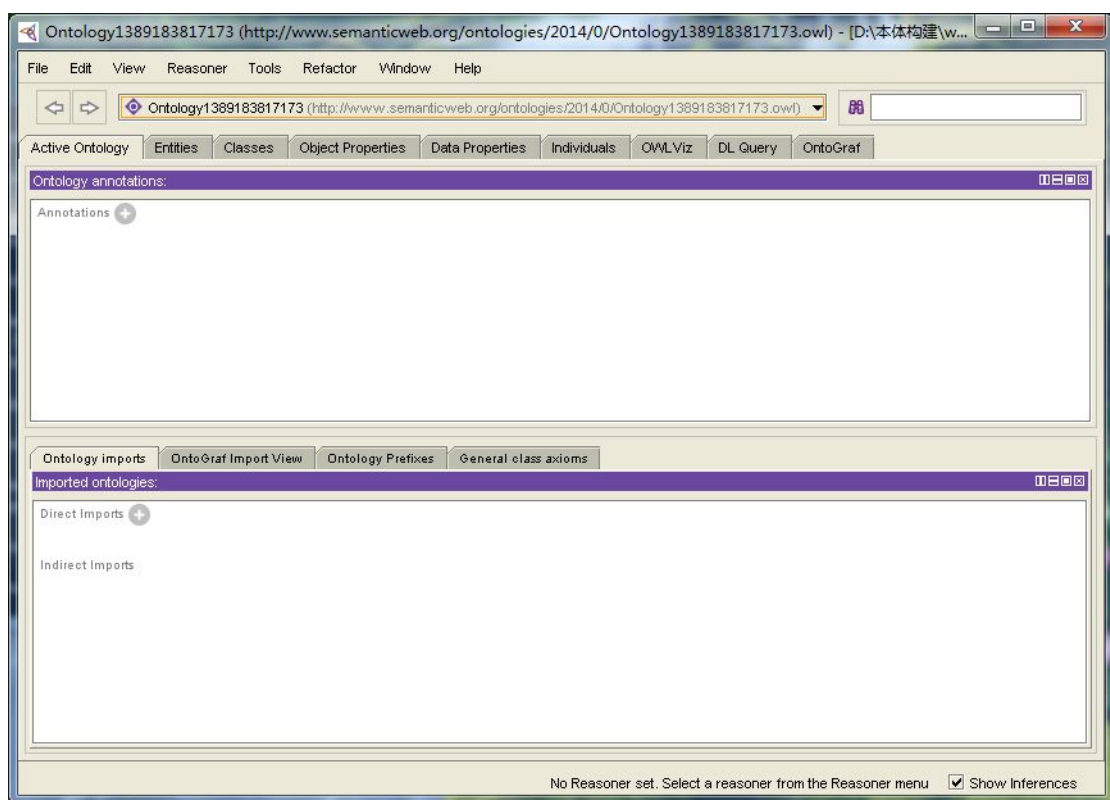


图 24

2、选择【Classes】标签，创建基本类层级结构，同时对各个类进行【Annotation】注释。Thing 有子类 Animal 和 Plant，Animal 有子类 Herbivore、Carnivore、Lion、Giaffe，Plant 有子类 Tree、Branch 和 Leaf。如图 25 所示。



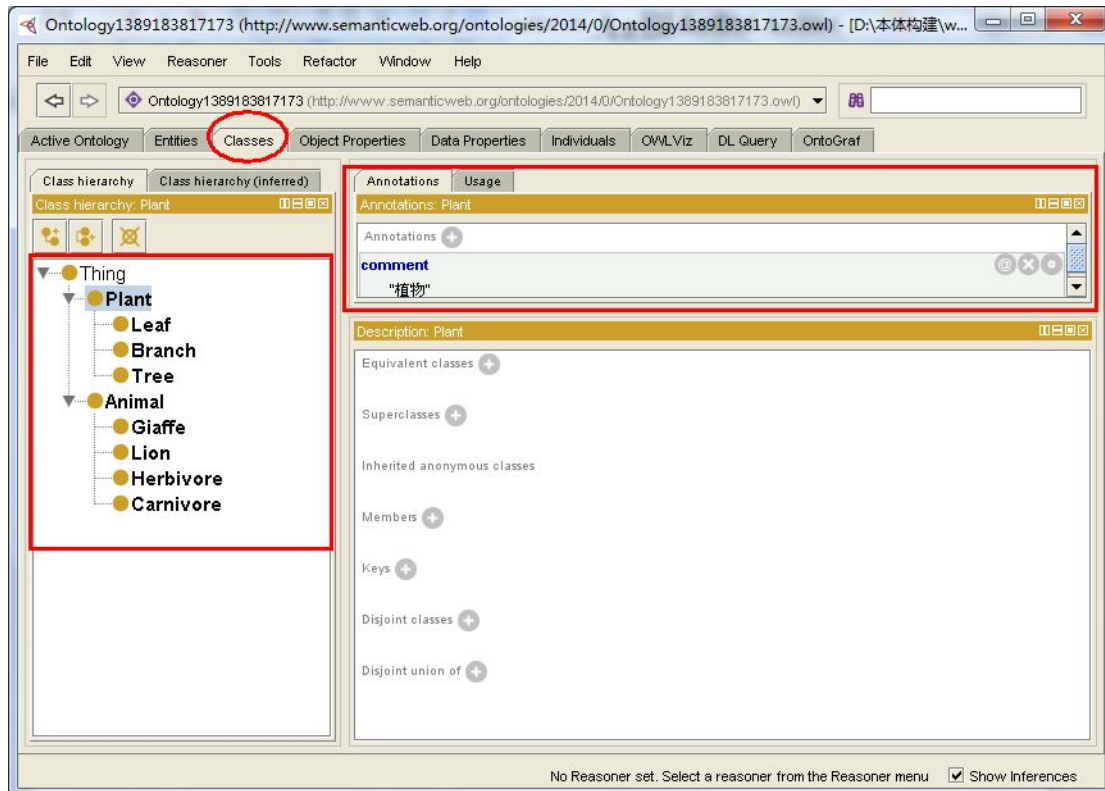


图 25

3、选择【Object Properties】标签，创建基本事物关联层级结构，并对各个关联进行【Annotation】注释。添加 topObjectProperty 的子类 eat、eated 和 is\_part\_of，eat 有子类 maineat。如图 26 所示。

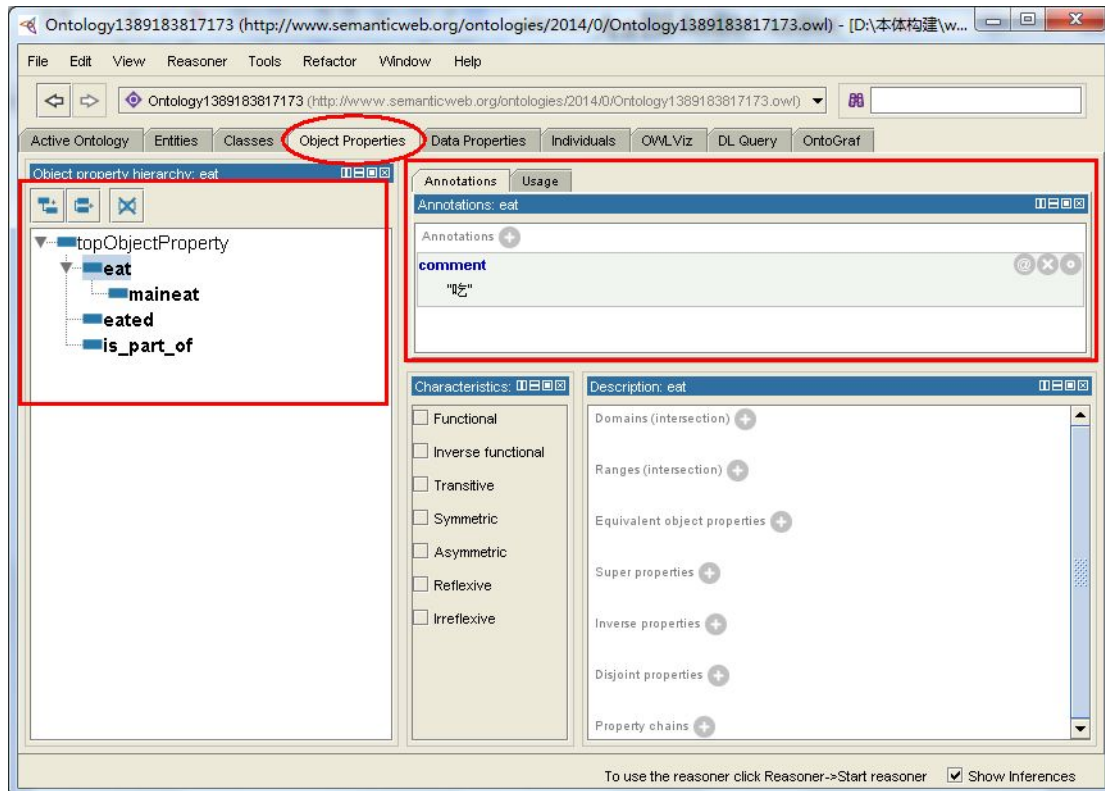


图 26

4、在【Object Properties】标签窗口设置关联属性。其中 is\_part\_of 关联具有传递性（【Characteristics】小标签中 Transitive 属性），同时设置 eat 关联和 eated 关联互为逆关联（【Description】小标签中的 Inverse properties 属性），并且 maineat 关联的值域为 Animal 类（【Description】小标签中的 Ranges 属性）。如图 27 所示。

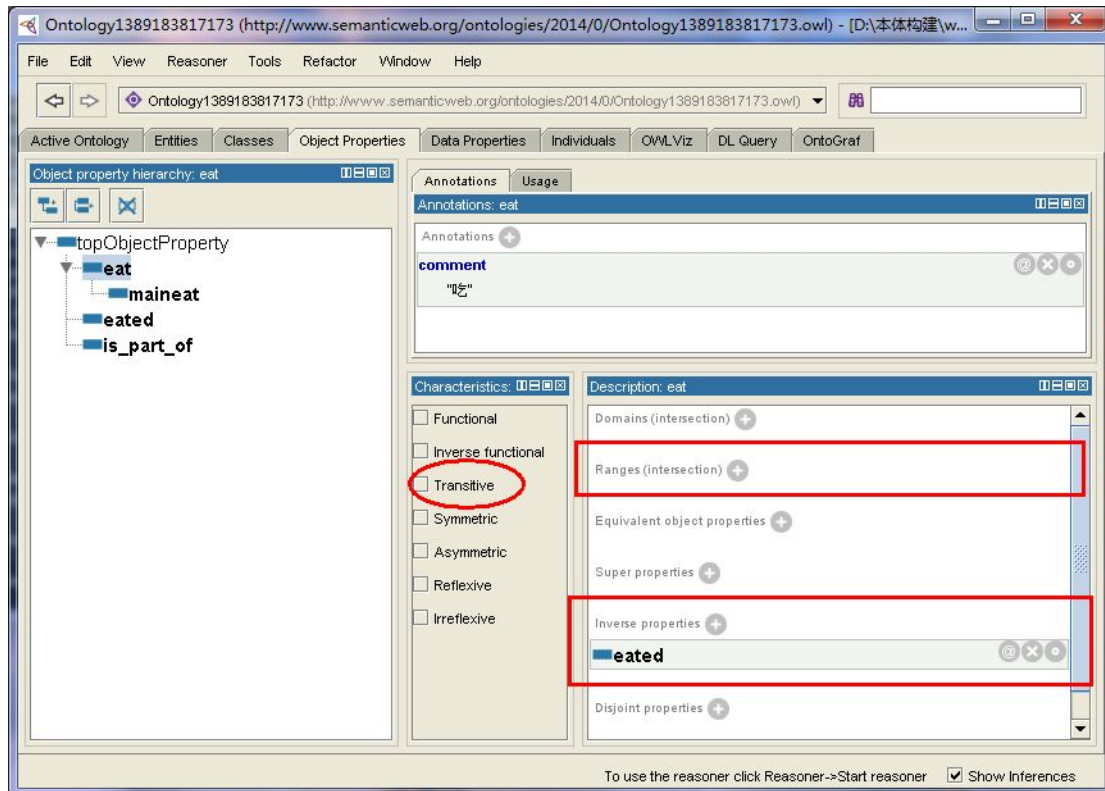


图 27

5、回到【Classes】标签，建立类与类之间的关系。设置①Animal类与Plant类不相交,即Animal类的Disjoint classes属性为Plant, ②Carnivore类的Equivalent classes属性为Animal and eat some Animal, ③Giaffe类的Superclasses为eat only Leaf, ④Herbivore类的Equivalent classes属性为Animal and eat only Plant, ⑤Lion类的Superclasses为maineat some Herbivore, 也可以把Lion类的事物关联“maineat”和Carnivore类的事物关联“eat”都设置为min, 其中Lion类的“Cardinality m (基数)”属性值m大于Carnivore类的事物关联“eat”的“Cardinality n”属性值n, ⑥Branch类的Superclasses为is\_part\_of only Tree, ⑦Leaf类的Superclasses为is\_part\_of only Branch。这样我们就设定了以下

事物关联规则：树叶是树枝的一部分，树枝是树的一部分，只吃植物的动物是食草动物，长颈鹿只吃树叶，狮子有时吃食草动物，可以吃动物的动物是食肉动物。如图 28 所示。

备注：由于中文逻辑比英文逻辑复杂，可以默认为 **protege** 中父类【对象】使用父类【事物关联】定义，子类【对象】使用子类【事物关联】定义。

举一个简单的例子：动物分为人和其它动物，会开车的是人，不会开车的是动物，会开好车的是土豪。“会开好车”是“开车”的子类，“土豪”是“人”的子类。

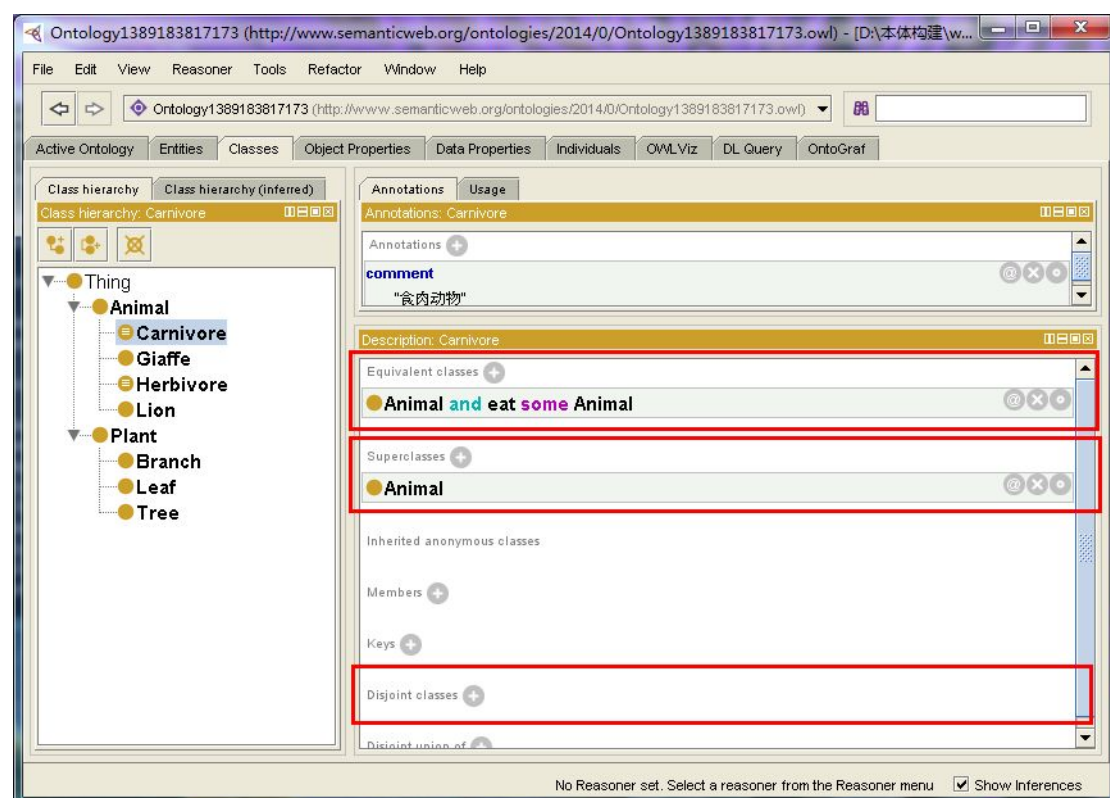


图 28

6、以上就设定好了本体类，类与类之间的关联及其规则，就可以查看本体关系图形，如图 29 所示。

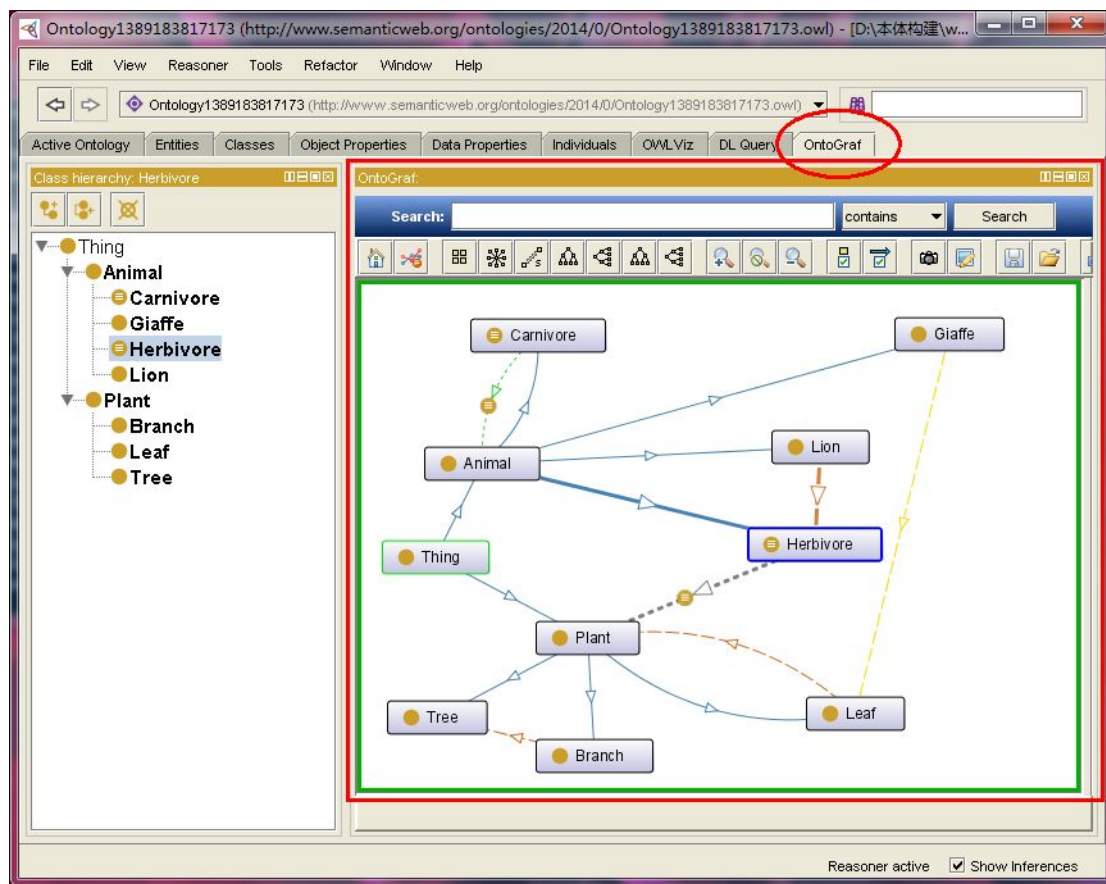
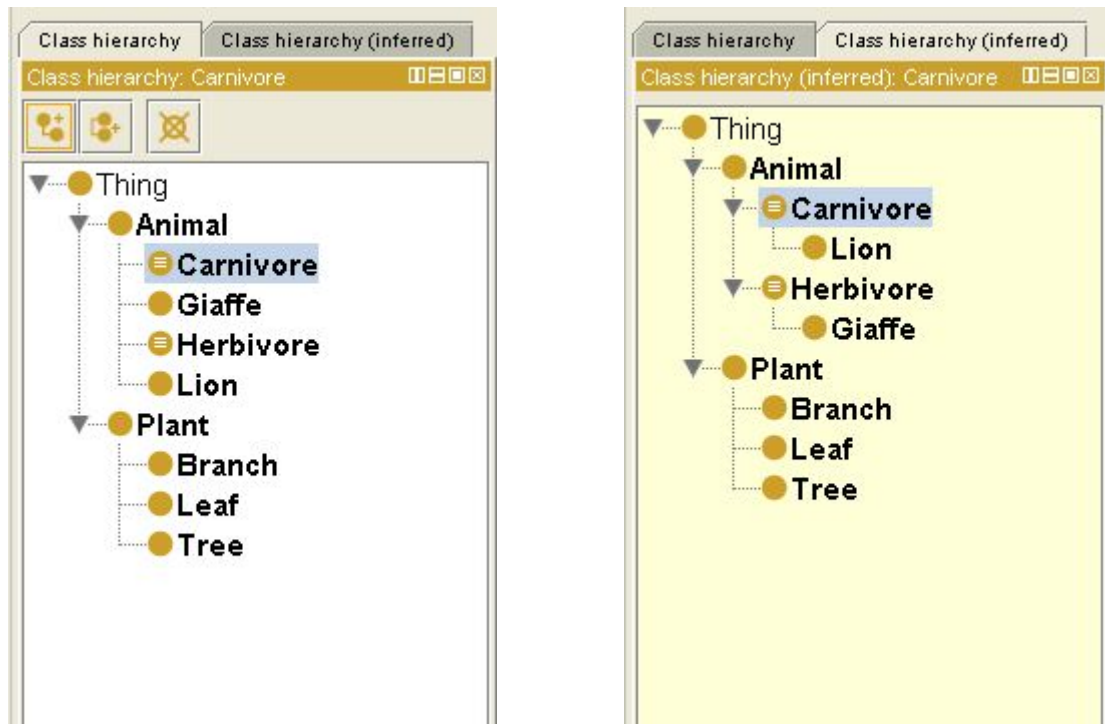


图 29

6、加载并且启动推理机，就可以进行推理、查询、可视化工作。

①类层级结构变化（左图是推理之前定义的，右图是推理之后的）

由此可见，Lion 类已经被推理为 Carnivore 类的子类，同时 Giaffe 类也被推理为 Herbivore 类的子类，这符合了现实的逻辑关系。



②本体可视化树形图变化（上图是推理之前形成的，下图是推理之后形成的）

备注：如果读者的 OWL Viz 功能因报错而无法使用的话，那是因为下载的 Protege 软件没自带 Graphviz 作图插件，读者可以自行下载和 Protege 版本匹配的 Graphviz 插件，然后打开菜单 Reasoner → configure 选项，再选择 OWL Viz 标签，设置 dot Application Path 路径值为你安装的 Graphviz 插件的安装路径 `\graphviz\bin\dot.exe`，也可以使用 Protege 自带的更新插件功能，这样 Protege 就会自动下载匹配的 Graphviz 插件，就可以直接使用 Protege 中 OWL Viz 功能。



