基于GOPPRR-E本体的KARMA模型信息读取规范

编制单位：北京理工大学，北京航空航天大学，中科蜂巢科技有限公司

版 本 号：第一版

二零二三年九月

编制		**生效日期**	
审核		**批准**	

文件变更摘要

日期	版本	变更说明	修订	审核	批准
2023-09-28	V0.1	第一版	鲁金直	鲁金直	鲁金直

目录

1 概述

1.1 目的

为了能够清晰的展示出如何从本体中获取模型信息 ,本文档以 A1 模型为例，通过 MetaGraph 工具导出 Karma.owl 文件 ,通过使用 Protégé工具生成 MetaGraph 模型的 GOPPRR-E 本体，获取本体模型中对应的模型信息。具体内容如下：

通过 MetaGraph 导出 Karma.owl 文件的步骤如下：

项目上右键选择导入 ,在打开的页面上选择 OWL 本体下面的 KARMA OWL 本体，点击下一步，并选择需要导出的项目，输入文件名以及想要导出的位置。

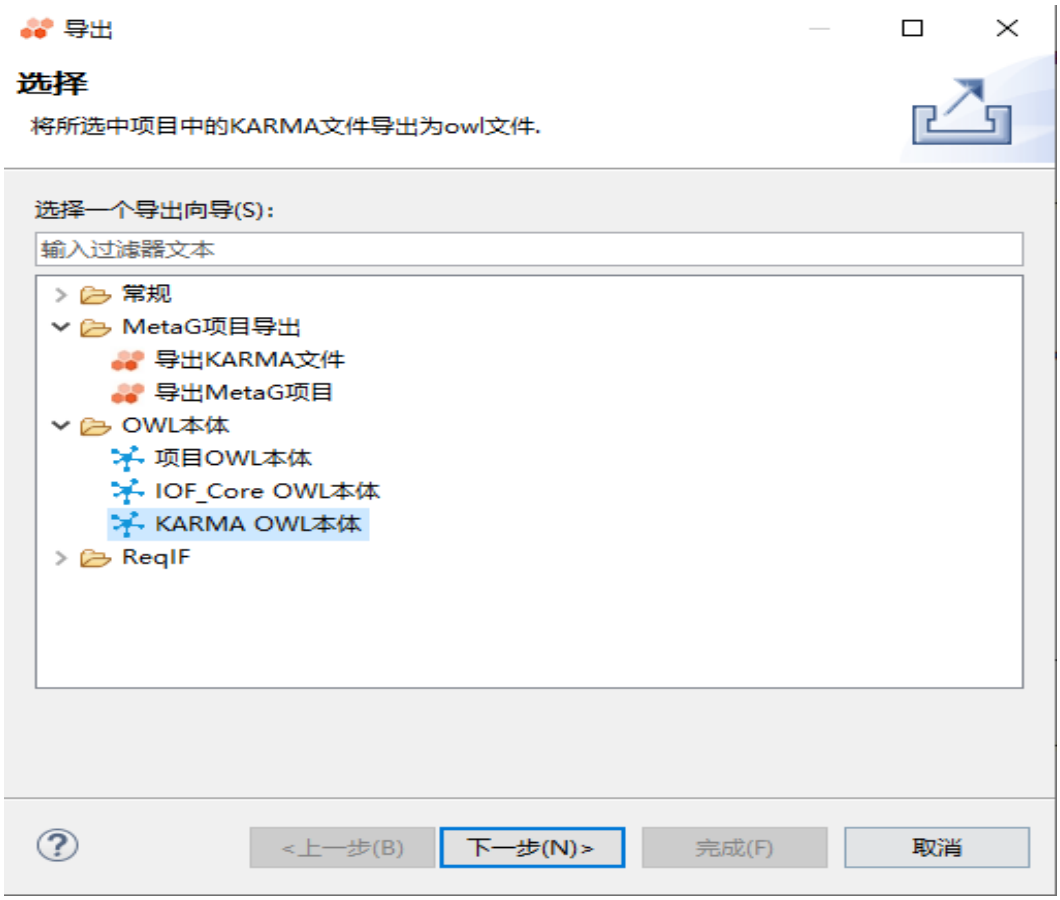

图 1-1导出KARMA.OWL本体（MetaGrap）

导出的 owl 文件使用 Protégé工具打开，生成 GOPPRR-E 本体。

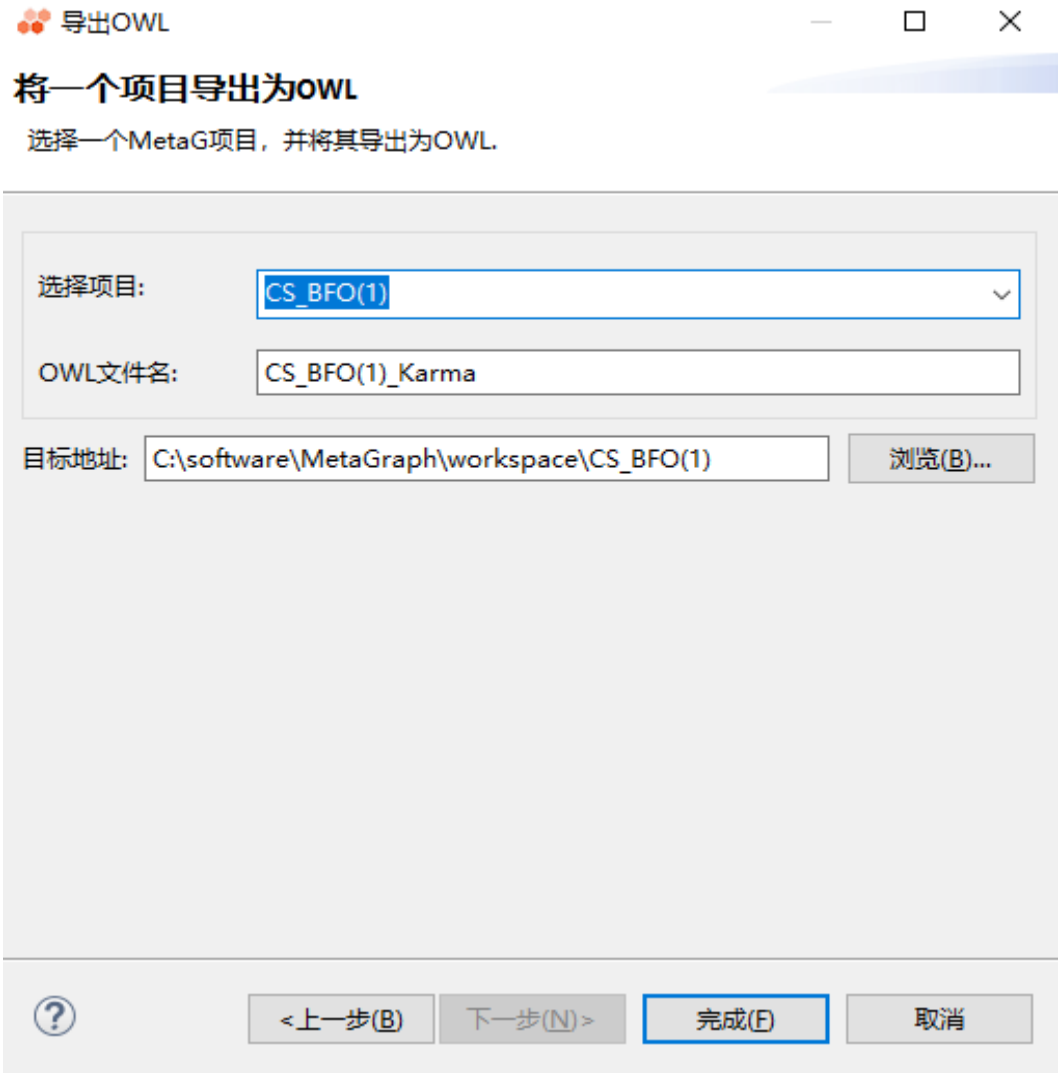

图 1-2导出KARMA.OWL本体（MetaGrap）

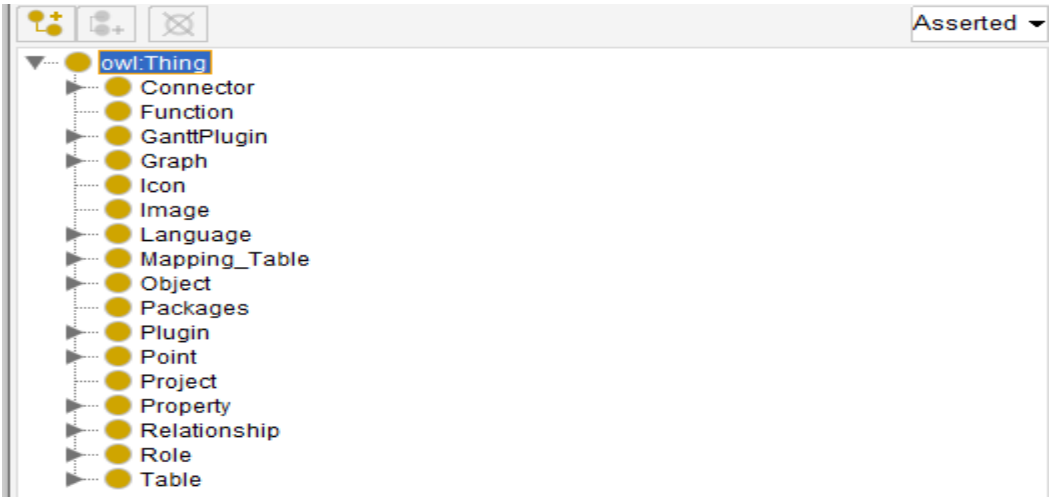

图 1-3GOPPRR_E（本体）

1.2 范围

本文档主要描述如何在本体中获取模型信息。对于如何获取某个具体的模型信息，本文档做了较为深入的阐述。

1.3 对应利益相关人

项目组成员：

需求方：

1.4 术语

表 1-1 术语

序号	术语	解释
1	Protege	用于展示本体GOPPRR_E的软件
2	MetaGraph	建模软件

2 方法及作用

方法名称	作用
getLanguage	从本体中获取语言信息
getMetaProperty	从本体中获取属性元模型信息
getMetaPoint	从本体中获取点元模型信息
getMetaRole	从本体中获取角色元模型信息
getMetaObject	从本体中获取对象元模型信息
getMetaRelationship	从本体中获取关系元模型信息

getMetaGraph	从本体中获取图元模型信息
getModel	从本体中获取模型信息

3 获取语言信息

3.1 目的

主要目的是处理本体数据，从指定 OWL 文件中读取语言信息，包括语言的 id,locallabel 等，具体信息如下：

（1） getLanguage该方法是一个静态公共方法，用于接收一个文件路径path作为参数;

（2） 声明METAG变量用于定义一个命名空间（namespace）的URL，用于在本体中唯一标识实体、属性等元素;

（3） 创建 OntModel 类型的本体模型对象 ontModel, 使用了 OntModelSpec.Owl_Men规范,同时将本体模型设置为非严格模式,避免读取实例时报错;

（4） 通过模型调用read方法，将具体的本体数据加载到ontModel对象中，供后续操作;

（5） 通过调用getAnnotationProperty方法，获取两个本体注释属性locallabel和id,用于描述语言实体;

（6） 通过调用getOntClass方法，获取本体中的Language类，表示MetaGraph中的语言文件夹;

（7） 循环遍历Language类的子类（建模语言），并从每个子类中提取出id

和localabel属性的值;

3.2 代码

```
1.   // 获取语言信息
2.  public static void getLanguage(String path) {
3.   // 命名空间（namespace）的URL，用于在本体建模中为实体、属性和其他元素提供唯一的标识符
4.   String METAG = "http://www.zkhoneycomb.com/formats/metagInOwl#";
5.   // 创建一个OntModel 对象
6.   OntModel ontModel = ModelFactory.createOntologyModel(OntModelSpec.OWL_MEM);
7.   // 从指定的文件路径(path)读取本体数据并将其加载到ontModel 对象中。
8.   ontModel.read(path);
9.   // 将ontModel 设置为非严格检测，这样由实例读取类时不会报错
10.  ontModel.setStrictMode(false);
11.  // 读取本体的注释属性
12.  AnnotationProperty localLabel = ontModel.getAnnotationProperty(METAG + "localLabel");
13.  AnnotationProperty id = ontModel.getAnnotationProperty(METAG + "id");
14.  // 读取本体中的Language 类，即MetaGraph 中的语言文件夹
15.  OntClass language = ontModel.getOntClass(METAG + "Language");
16.  // 遍历本体中的Language 类下的子类，即建模语言
17.  for (ExtendedIterator<OntClass> it = language.listSubClasses(); it.hasNext();) {
18.   OntClass ontLanguage = it.next();
19.   String ontLanguageId = ontLanguage.getPropertyValue(id).toString();
20.   String ontLanguageName = ontLanguage.getPropertyValue(localLabel).toString();
21.   System.out.println(ontLanguageId);
22.   System.out.println(ontLanguageName);
23.  }
24. }
```

3.3 案例

通过调用getLanguage方法获取的信息如下所示：

KKsWVy0U1即为ontLanguageId 。

（1） Protégé工具：

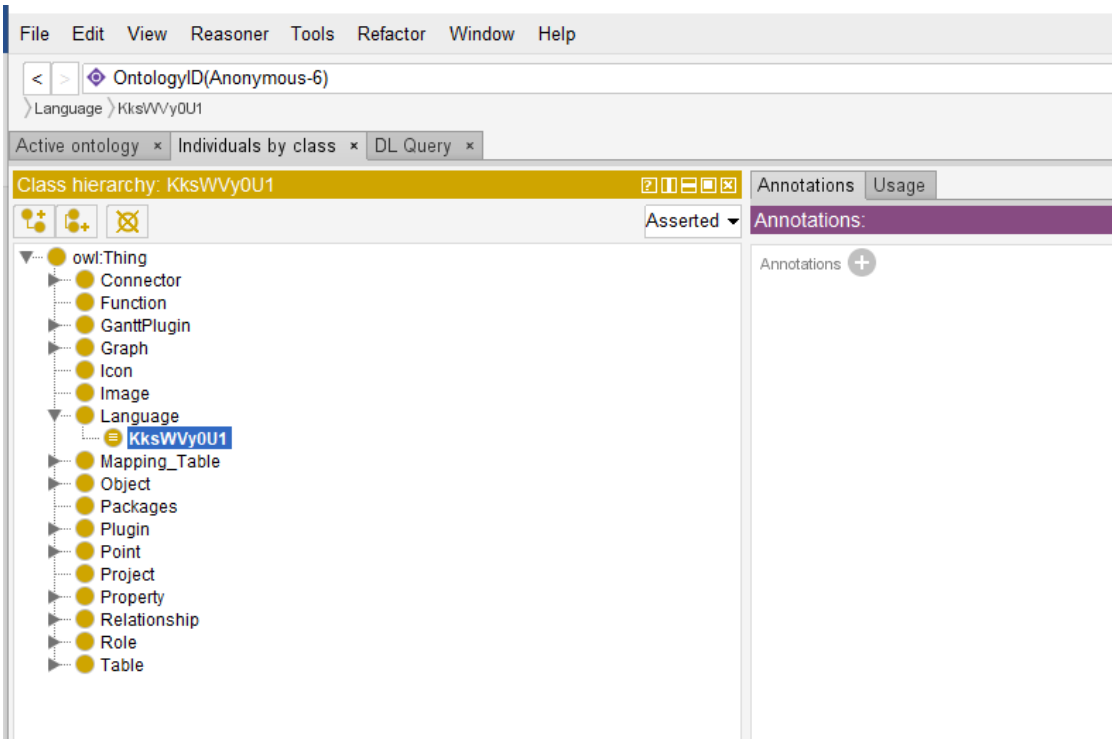

图 3-1语言信息（本体）

（2） MetaGraph工具：

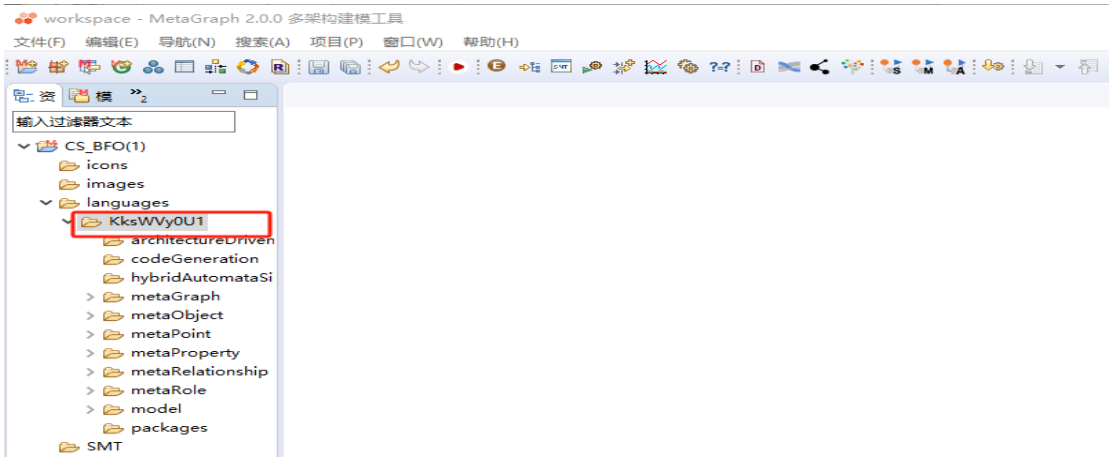

图 3-2语言信息（MetaGraph）

4 获取属性元模型信息

4.1 目的

主要目的是处理本体数据,从指定OWL文件中读取属性元模型信息,获取属性元模型的id，以及localabel等属性，具体信息如下：

（1） getMetaProperty该方法是一个静态公共方法,用于接收一个文件路径path作为参数;

（2） 声明METAG变量用于定义一个命名空间（namespace）的URL，用于在本体中唯一标识实体、属性等元素;

（3） 创建OntModel类型的本体模型对象ontModel,使用了OntModelSpec.Owl_Men规范,同时将本体模型设置为非严格模式,避免读取实例时报错;

（4） 通过调用read方法，将具体的本体数据加载到ontModel对象中，供后续操作;

（5） 通过调用getAnnotationProperty方法，获取本体注释属性localLabel,annotationPropertyDataType等;

（6） 通过调用getOntClass方法，获取本体中的Language类，表示MetaGraph中的语言文件夹;

（7） 通过调用getObjectProperty方法，获取本体的对象属性，即元素之间的关系；

（8） 通过嵌套的循环遍历了 Language 类的子类以及与这些子类等价的类；

（9） 在循环中，判断等价类是否具有属性 languageIncludingProperty，如果是，则进一步获取这个建模语言的属性元模型信息。包括属性元模型id,属性元模型的locallabel, 属性元模型的注释属性description等属性值；

4.2 代码

```
1.  // 获取属性元模型信息
2.  public static void getMetaProperty(String path) {
3.   // 命名空间（namespace）的URL，用于在本体建模中为实体、属性和其他元素提供唯一的标识符
4.   String METAG = "http://www.zkhoneycomb.com/formats/metagInOwl#";
5.   // 创建一个OntModel 对象
6.   OntModel ontModel = ModelFactory.createOntologyModel(OntModelSpec.OWL_MEM);
7.   // 从指定的文件路径(path)读取本体数据并将其加载到ontModel 对象中。
8.   ontModel.read(path);
9.   // 读取本体的注释属性
10.  AnnotationProperty localLabel = ontModel.getAnnotationProperty(METAG + "localLabel");
11.  AnnotationProperty annotationPropertyDataType = ontModel.getAnnotationProperty(METAG + "dataType");
12.  AnnotationProperty annotationPropertyUnit = ontModel.getAnnotationProperty(METAG + "unit");
13.  AnnotationProperty description = ontModel.getAnnotationProperty(METAG + "description");
14.
15.  // 读取本体中的Language 类，即MetaGraph 中的语言文件夹
16.  OntClass language = ontModel.getOntClass(METAG + "Language");
17.  // 读取本体的对象属性，即元素之间的关系
18.  ObjectProperty languageIncludingProperty = ontModel.getObjectProperty(METAG + "languageIncludingProperty");
19.  // 遍历本体中的Language 类下的子类，即建模语言
20.  for (ExtendedIterator<OntClass> it = language.listSubClasses(); it.hasNext();) {
21.   OntClass ontLanguage = it.next();
22.   // 遍历本体中的Language 子类的等价类，即与子类有关系的类
23.   for (ExtendedIterator<OntClass> it2 = ontLanguage.listEquivalentClasses(); it2.hasNext();) {
24.    OntClass equivalentClass = it2.next();
25.    // 判断等价类是否具有属性 languageIncludingProperty
```

```
26.     if (equivalentClass.asRestriction().getOnProperty().equals(la
    nguageIncludingProperty)) {
27.         // 将等价类具有的属性 languageIncludingProperty 的属性值取出来，
    即获取这个建模语言具有的属性元模型
28.         OntClass ontMetaProperty = (OntClass) equivalentClass.asRest
    riction().asSomeValuesFromRestriction()
29.                 .getSomeValuesFrom();
30.         // 获取属性元模型的 Id
31.         String metaPropertyName = ontMetaProperty.getLocalName();
32.         // 获取属性元模型的 localLabel
33.         String metaPropertyLocalLabel = ontMetaProperty.getPropertyV
    alue(localLabel).toString();
34.         // 获取属性元模型的注释属性 description 值(*)
35.         if (ontMetaProperty.getPropertyValue(description) != null) {
36.             String metaPropertyDescription = ontMetaProperty.getPropert
    yValue(description).toString();
37.         }
38.         // 获取属性元模型的数据类型 dataType 值(*)
39.         String metaPropertyDataType = ontMetaProperty.getPropertyVal
    ue(annotationPropertyDataType)
40.                 .toString();
41.         // 获取属性元模型的注释属性 unit 值(*)
42.         String metaPropertyUnit;
43.         if (ontMetaProperty.hasProperty(annotationPropertyUnit)) {
44.             metaPropertyUnit = ontMetaProperty.getPropertyValue(annotat
    ionPropertyUnit).toString();
45.         }
46.     }
47.   }
48.  }
49. }
```

4.3 案例

通过调用 getMetaProperty 方法获取的信息如下所示：

其中 Message_Sort 即为 metaPropertyName，也就是属性元模型的 id，消息类为其 metaPropertyLocalLabel，也就是属性元模型的 locallabel。

（1） Protégé工具

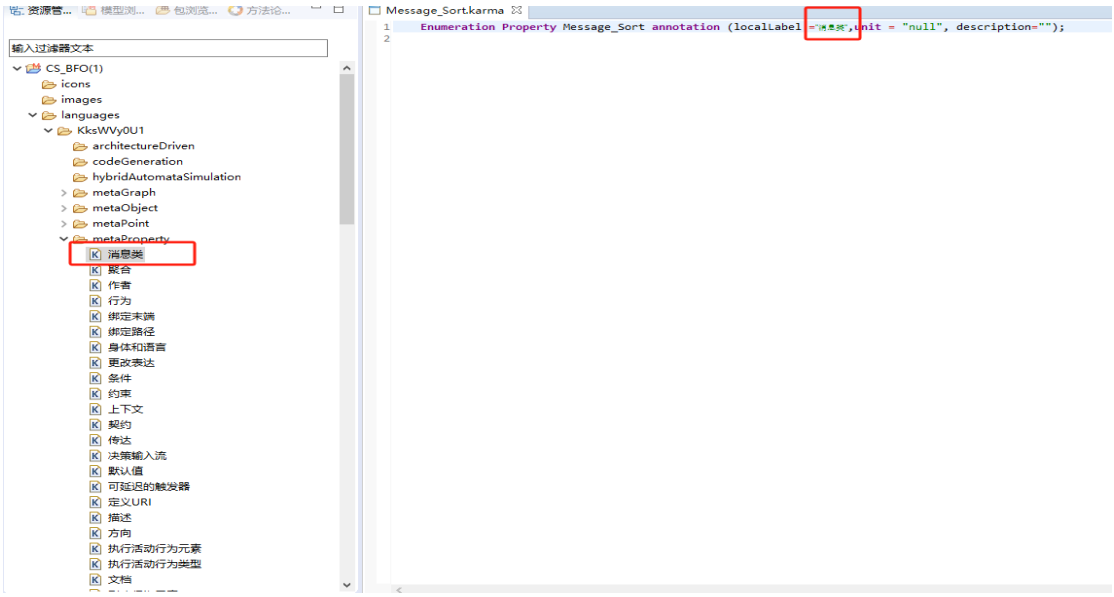

图 4-1属性元模型信息（本体）

（2） MetaGraph工具

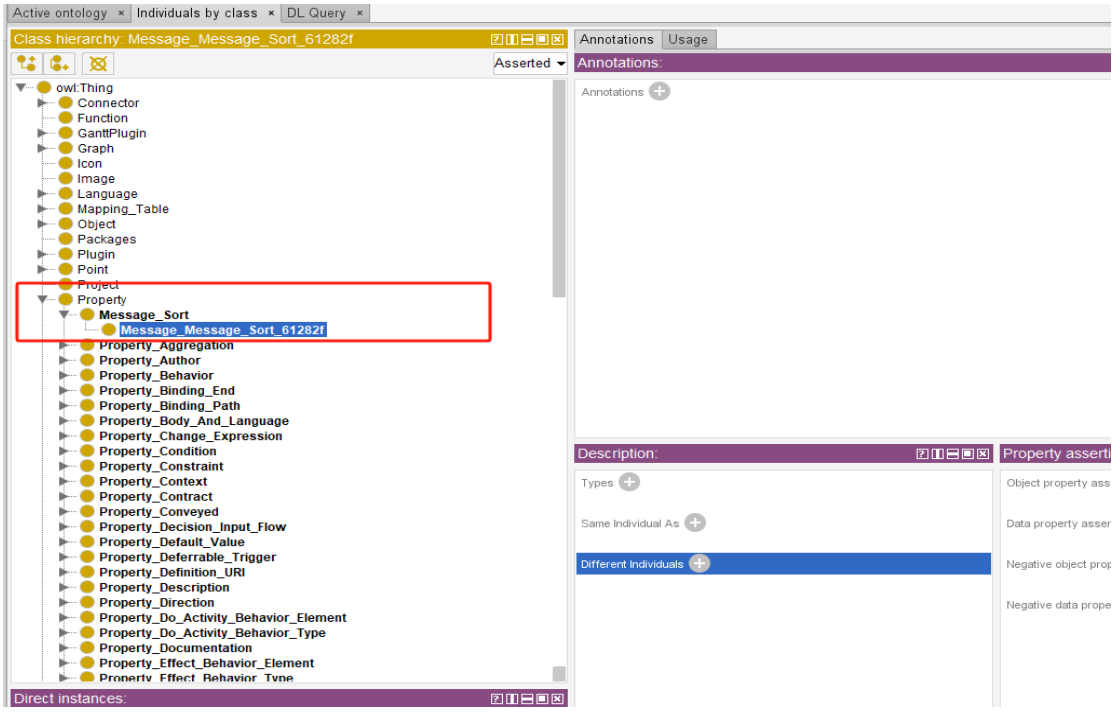

图 4-2属性元模型信息（MetaGraph）

5 获取点元模型信息

5.1 目的

主要目的是处理本体数据,从指定OWL文件中读取点元模型信息,具体信息如下：

（1） getMetaPoint该方法是一个静态公共方法，用于接收一个文件路径path作为参数;

（2） 声明METAG变量用于定义一个命名空间（namespace）的URL，用于在本体中唯一实体标识、属性等元素;

（3） 创建OntModel类型的本体模型对象ontModel, 使用了OntModelSpec.Owl_Men规范,同时将本体模型设置为非严格模式,避免读取实例时报错;

（4） 通过调用read方法，将具体的本体数据加载到ontModel对象中，供后续操作;

（5） 通过调用getAnnotationProperty方法，获取本体注释属性localLabel, description,shape等;

（6） 通过调用getOntClass方法，获取本体中的Language类，表示MetaGraph中的语言文件夹;

（7） 通过调用getObjectProperty方法获取本体的对象属性languageIncludingPoint，该属性表示元素之间的关系，语言包括的点元模型。

（8） 通过嵌套的循环遍历了 Language 类的子类以及与这些子类等价的类；

（9） 在循环中，代码判断等价类是否具有属性languageIncludingPoint，如果是，则进一步获取该建模语言的点元模型信息。包括id，localabel等

（10）同时在内部循环中，获取点元模型的属性元模型信息，通过遍历ontMetaPoint的等价类列表，获取具有关系的hasProperty属性元模型（metaPointProperty）

5.2 代码

```
1.  // 获取点元模型信息
2.  public static void getMetaPoint(String path) {
3.   // 命名空间（namespace）的URL，用于在本体建模中为实体、属性和其他元素提供唯一的标识符
4.   String METAG = "http://www.zkhoneycomb.com/formats/metagInOwl#";
5.   // 创建一个OntModel对象
6.   OntModel ontModel = ModelFactory.createOntologyModel(OntModelSpec.OWL_MEM);
7.   // 从指定的文件路径(path)读取本体数据并将其加载到ontModel对象中。
8.   ontModel.read(path);
9.   // 读取本体的注释属性
10.  AnnotationProperty localLabel = ontModel.getAnnotationProperty(METAG + "localLabel");
11.  AnnotationProperty description = ontModel.getAnnotationProperty(METAG + "description");
12.  AnnotationProperty shape = ontModel.getAnnotationProperty(METAG + "shape");
13.  AnnotationProperty id = ontModel.getAnnotationProperty(METAG + "id");
14.  // 读取本体中的Language类，即MetaGraph中的语言文件夹
15.  OntClass language = ontModel.getOntClass(METAG + "Language");
16.  // 读取本体的对象属性，即元素之间的关系，语言包括的点元模型
17.  ObjectProperty languageIncludingPoint = ontModel.getObjectProperty(METAG + "languageIncludingPoint");
18.  // 遍历本体中的Language类下的子类，即建模语言
19.  for (ExtendedIterator<OntClass> it = language.listSubClasses(); it.hasNext();) {
20.   OntClass ontLanguage = it.next();
21.   String ontLanguageId = ontLanguage.getPropertyValue(id).toString();
```

```
22.   String ontLanguageName = ontLanguage.getPropertyValue(localLab
   el).toString();
23.
24.   for (ExtendedIterator<OntClass> it2 = ontLanguage.listEquivale
   ntClasses(); it2.hasNext();) {
25.    OntClass equivalentClass = it2.next();
26.    if (equivalentClass.asRestriction().getOnProperty().equals(la
   nguageIncludingPoint)) {
27.     OntClass ontMetaPoint = (OntClass) equivalentClass.asRestric
   tion().asSomeValuesFromRestriction()
28.       .getSomeValuesFrom();
29.     // 获取点元模型的 ID 属性
30.     String metaPointName = ontMetaPoint.getLocalName();
31.     // 获取点元模型的 localLabel 属性
32.     String metaPointLocalLabel = ontMetaPoint.getPropertyValue(l
   ocalLabel).toString();
33.     // 获取点元模型的 description 属性
34.     if (ontMetaPoint.getPropertyValue(description) != null) {
35.      String metaPointDescription = ontMetaPoint.getPropertyValue
   (description).toString();
36.     }
37.     // 获取点元模型的形状属性
38.     String metaPointShape = ontMetaPoint.getPropertyValue(shape)
   .toString();
39.
40.     // 点元模型具有的属性元模型
41.     for (Iterator point_it = ontMetaPoint.listEquivalentClasses(
   ); point_it.hasNext();) {
42.      OntClass c2 = (OntClass) point_it.next();
43.      Restriction r2 = c2.asRestriction();
44.
45.      if (r2.isSomeValuesFromRestriction()
46.        && r2.getOnProperty().getLocalName().equals("hasProperty"
   )) {
47.       SomeValuesFromRestriction sr2 = r2.asSomeValuesFromRestric
   tion();
48.       OntClass metaPointProperty = (OntClass) sr2.getSomeValuesF
   rom();
49.      }
50.
51.      if (r2.isAllValuesFromRestriction()
52.        && r2.getOnProperty().getLocalName().equals("hasProperty"
   )) {
```

```
53.         AllValuesFromRestriction sr2 = r2.asAllValuesFromRestricti
    on();
54.         OntClass metaPointProperty = (OntClass) sr2.getAllValuesFr
    om();
55.
56.       }
57.      }
58.     }
59.    }
60.
61.   }
62. }
```

5.3 案例

通过调用 getMetaPoint 方法获取的信息如下所示：

其中 Point_Action_Input_Pin 为 metaPointName 也就是点元模型的 id 属性，动作输入引脚为 metaPointLocalLabel 也就是点元模型的 locallabel 属性，Point_Action_Input_Pin_Property_From_Action_c9e7b6 等为点元模型的属性元模型信息。

（1） Protégé工具

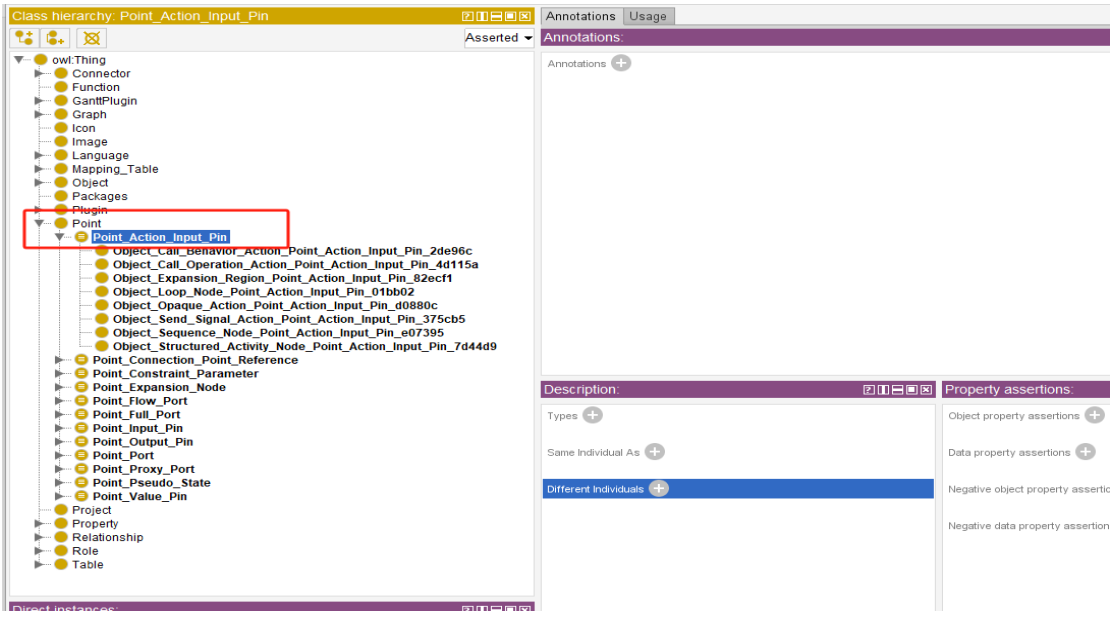

图 5-1点元模型信息（本体）

（2） MetaGraph工具

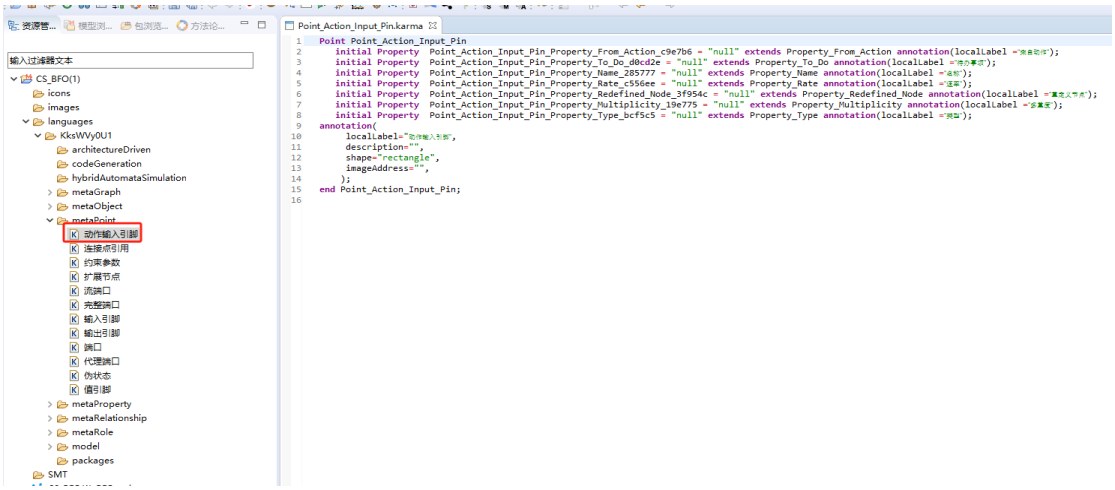

图 5-2点元模型信息（MetaGraph）

（3） 点元模型的属性元模型

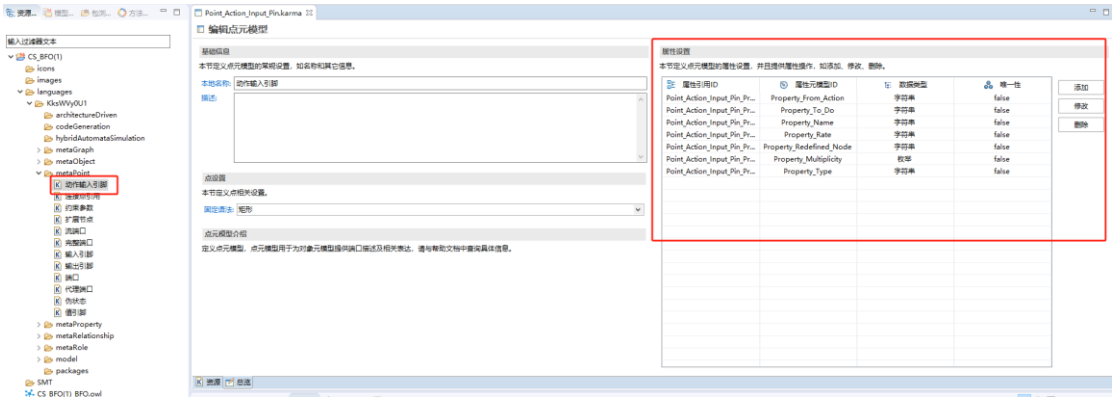

图 5-3点元模型的属性元模型信息（MetaGraph）

6 获取角色元模型信息

6.1 目的

主要目的是处理本体数据 ,从指定OWL文件中读取角色元模型信息 ,具体信息如下：

（1） getMetaRole该方法是一个静态公共方法，用于接收一个文件路径

path作为参数;

（2） 声明METAG变量用于定义一个命名空间（namespace）的URL，用于在本体中唯一实体标识、属性等元素;

（3） 创建OntModel类型的本体模型对象ontModel, 使用了OntModelSpec.Owl_Men规范,同时将本体模型设置为非严格模式,避免读取实例时报错;

（4） 通过调用read方法，将具体的本体数据加载到ontModel对象中，供后续操作;

（5） 通过调用getAnnotationProperty方法，获取本体注释属性localLabel, description,shape等;

（6） 通过调用getOntClass方法，获取本体中的Language类，表示MetaGraph中的语言文件夹;

（7） 通过调用getObjectProperty方法，获取本体的对象属性languageIncludingRole，该属性表示元素之间的关系，语言包括角色元模型;

（8） 通过嵌套的循环遍历了 Language 类的子类以及与这些子类等价的类;

（9） 在循环中，代码判断等价类是否具有属性languageIncludingRole，如果是，则进一步获取该建模语言角色元模型信息，包括id,localabel等;

（10）同时在内部循环中,获取角色元模型具有的属性元模型信息,通过遍历ontMetaRole的等价类列表，获取具有关系的hasProperty属性元模型（metaRoleProperty）;

6.2 代码

```
1.  // 获取角色元模型信息
2.  public static void getMetaRole(String path) {
3.   // 命名空间（namespace）的URL，用于在本体建模中为实体、属性和其他元素提供唯一的标识符
4.   String METAG = "http://www.zkhoneycomb.com/formats/metagInOwl#";
5.   // 创建一个OntModel 对象
6.   OntModel ontModel = ModelFactory.createOntologyModel(OntModelSpec.OWL_MEM);
7.   // 从指定的文件路径(path)读取本体数据并将其加载到ontModel 对象中。
8.   ontModel.read(path);
9.   // 读取本体的注释属性
10.  AnnotationProperty localLabel = ontModel.getAnnotationProperty(METAG + "localLabel");
11.  AnnotationProperty description = ontModel.getAnnotationProperty(METAG + "description");
12.  AnnotationProperty shape = ontModel.getAnnotationProperty(METAG + "shape");
13.  AnnotationProperty direction = ontModel.getAnnotationProperty(METAG + "direction");
14.  AnnotationProperty id = ontModel.getAnnotationProperty(METAG + "id");
15.
16.  OntClass language = ontModel.getOntClass(METAG + "Language");
17.  ObjectProperty languageIncludingRole = ontModel.getObjectProperty(METAG + "languageIncludingRole");
18.
19.  for (ExtendedIterator<OntClass> it = language.listSubClasses(); it.hasNext();) {
20.   OntClass ontLanguage = it.next();
21.   String ontLanguageId = ontLanguage.getPropertyValue(id).toString();
22.   String ontLanguageName = ontLanguage.getPropertyValue(localLabel).toString();
23.
24.   for (ExtendedIterator<OntClass> it2 = ontLanguage.listEquivalentClasses(); it2.hasNext();) {
25.    OntClass equivalentClass = it2.next();
26.    if (equivalentClass.asRestriction().getOnProperty().equals(languageIncludingRole)) {
```

```
27.     OntClass ontMetaRole = (OntClass) equivalentClass.asRestrict
    ion().asSomeValuesFromRestriction()
28.       .getSomeValuesFrom();
29.     // 获取角色元模型的 ID 属性
30.     String metaRoleName = ontMetaRole.getLocalName();
31.     // 获取角色元模型的 localLabel 属性
32.     String metaRoleLocalLabel = ontMetaRole.getPropertyValue(loc
    alLabel).toString();
33.     // 获取角色元模型的 description 属性(*)
34.     if (ontMetaRole.getPropertyValue(description) != null) {
35.      String metaRoleDescription = ontMetaRole.getPropertyValue(d
    escription).toString();
36.     }
37.     // 获取角色元模型的 direction 属性
38.     String metaRoleDirection = ontMetaRole.getPropertyValue(dire
    ction).toString();
39.     // 获取角色元模型的 shape 属性
40.     String metaRoleShape = ontMetaRole.getPropertyValue(shape).t
    oString();
41.
42.     // 获取角色元模型具有的属性元模型
43.     for (Iterator role_it = ontMetaRole.listEquivalentClasses();
     role_it.hasNext();) {
44.      OntClass c2 = (OntClass) role_it.next();
45.      Restriction r2 = c2.asRestriction();
46.      // 表示 r2 是一个 "某个值来自" 限制，并且应用
    于 "hasProperty" 属性。
47.      if (r2.isSomeValuesFromRestriction()
48.        && r2.getOnProperty().getLocalName().equals("hasProperty"
    )) {
49.       SomeValuesFromRestriction sr2 = r2.asSomeValuesFromRestric
    tion();
50.       OntClass metaRoleProperty = (OntClass) sr2.getSomeValuesFr
    om();
51.      }
52.      // 表示 r2 是一个 "所有值来自" 限制，并且应用
    于 "hasProperty" 属性
53.      if (r2.isAllValuesFromRestriction()
54.        && r2.getOnProperty().getLocalName().equals("hasProperty"
    )) {
55.       AllValuesFromRestriction sr2 = r2.asAllValuesFromRestricti
    on();
56.       OntClass metaRoleProperty = (OntClass) sr2.getAllValuesFro
    m();
```

```
57.       }
58.      }
59.     }
60.    }
61.
62.   }
63. }
```

6.3 案例

通过调用 getMetaRole 方法获取的信息如下所示：

其中 Role_Role_B 为 metaRoleName 也就是角色元模型的 id 属性 ,角色 B 为 metaRoleLocalLabel 也就是角色元模型的 locallabel 属性, Property_Context 为角色元模型具有的属性元模型。

（1） Protégé工具

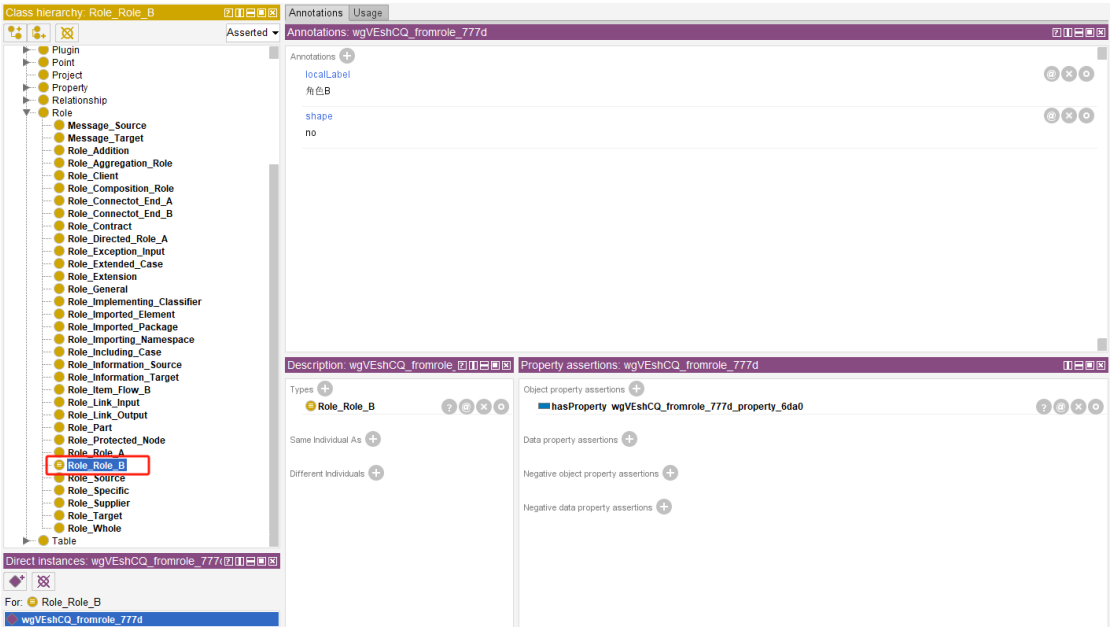

图 6-1角色元模型信息（本体）

（2） MetaGraph工具

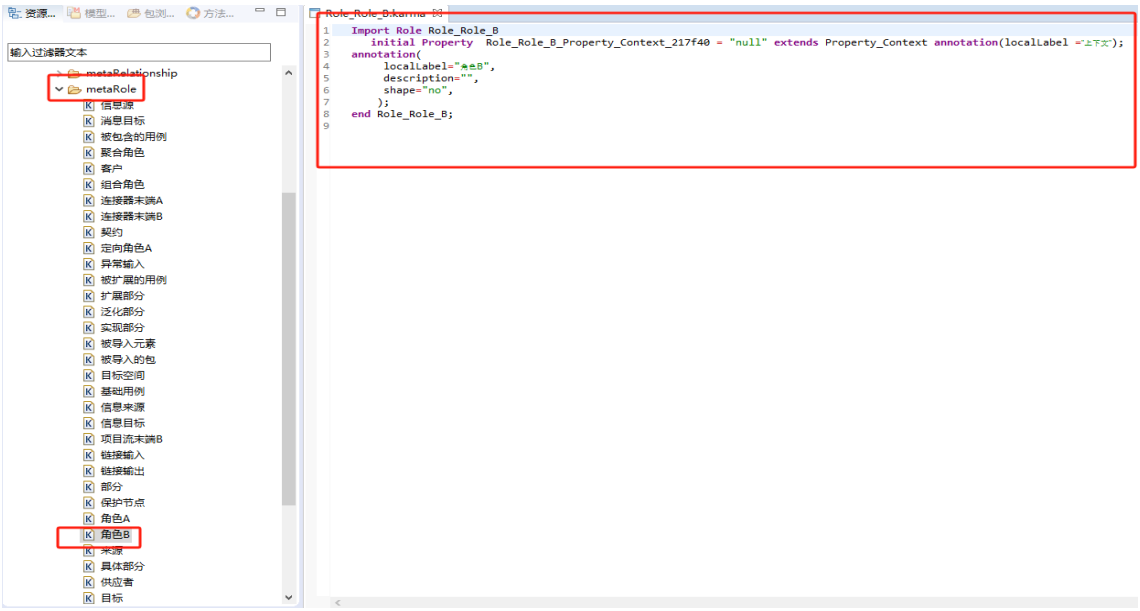

图 6-2角色元模型信息（MetaGraph）

7 获取对象元模型信息

7.1 目的

主要目的是处理本体数据，从指定OWL文件中读取对象元模型信息，具体信息如下：

（1） getMetaObject该方法是一个静态公共方法，用于接收一个文件路径path作为参数;

（2） 声明METAG变量用于定义一个命名空间（namespace）的URL，用于在本体中唯一实体标识、属性等元素;

（3） 创建OntModel类型的本体模型对象ontModel, 使用了OntModelSpec.Owl_Men规范,同时将本体模型设置为非严格模式,避免读取实例时报错;

（4） 通过调用read方法，将具体的本体数据加载到ontModel对象中，供后续操作;

（5） 通过调用getAnnotationProperty方法，获取本体注释属性localLabel, description,shape等,描述对象元模型的各种属性；

（6） 通过调用getOntClass方法，获取本体中的Language类，表示MetaGraph中的语言文件夹;

（7） 通过调用getObjectProperty方法，获取本体的对象属性languageIncludingObject，该属性表示元素之间的关系，语言包括的对象元模型;

（8） 通过嵌套的循环遍历了 Language 类的子类以及与这些子类等价的类；

（9） 在循环中,代码判断等价类是否具有属性languageIncludingObject,如果是，则进一步获取该建模语言的对象元模型信息，包括id，locallabel等；

（10）同时在内部循环中,获取对象元模型具有的属性元模型信息,通过遍历ontMetaObject的等价类列表，获取具有关系的hasProperty属性元模型（metaObjectProperty）;

7.2 代码

```
1.  // 获取对象元模型信息
2.  public static void getMetaObject(String path) {
3.    // 命名空间（namespace）的URL，用于在本体建模中为实体、属性和其他元素提供唯一的标识符
4.    String METAG = "http://www.zkhoneycomb.com/formats/metagInOwl#";
5.    // 创建一个OntModel对象
6.    OntModel ontModel = ModelFactory.createOntologyModel(OntModelSpec.OWL_MEM);
7.    // 从指定的文件路径(path)读取本体数据并将其加载到ontModel对象中。
8.    ontModel.read(path);
```

```
    // 读取本体的注释属性
    AnnotationProperty localLabel = ontModel.getAnnotationProperty(METAG + "localLabel");
    AnnotationProperty description = ontModel.getAnnotationProperty(METAG + "description");
    AnnotationProperty annotationPropertyScreenMode = ontModel.createAnnotationProperty(METAG + "screenMode");// visualizedMode
    AnnotationProperty annotationPropertyUnfoldDirection = ontModel
        .getAnnotationProperty(METAG + "UnfoldDirection");
    AnnotationProperty shape = ontModel.getAnnotationProperty(METAG + "shape");
    AnnotationProperty annotationPropertyIcon = ontModel.getAnnotationProperty(METAG + "icon");
    AnnotationProperty annotationPropertyImageAddress = ontModel.getAnnotationProperty(METAG + "imageAddress");
    AnnotationProperty id = ontModel.getAnnotationProperty(METAG + "id");

    OntClass language = ontModel.getOntClass(METAG + "Language");
    ObjectProperty languageIncludingObject = ontModel.getObjectProperty(METAG + "languageIncludingObject");

    for (ExtendedIterator<OntClass> it = language.listSubClasses(); it.hasNext();) {
      OntClass ontLanguage = it.next();
      String ontLanguageId = ontLanguage.getPropertyValue(id).toString();
      String ontLanguageName = ontLanguage.getPropertyValue(localLabel).toString();
      System.out.println(ontLanguageId + ", " + ontLanguageName);

      for (ExtendedIterator<OntClass> it2 = ontLanguage.listEquivalentClasses(); it2.hasNext();) {
        OntClass equivalentClass = it2.next();
        if (equivalentClass.asRestriction().getOnProperty().equals(languageIncludingObject)) {
          OntClass ontMetaObject = (OntClass) equivalentClass.asRestriction().asSomeValuesFromRestriction()
              .getSomeValuesFrom();
          // 获取对象元模型的 ID 属性
          String metaObjectName = ontMetaObject.getLocalName();
          // 获取对象元模型的 localLabel 属性
          String metaObjectLocalLabel = ontMetaObject.getPropertyValue(localLabel).toString();
```

```
38.     // 获取对象元模型的 description 属性
39.     if (ontMetaObject.getPropertyValue(description) != null) {
40.      String metaObjectDescription = ontMetaObject.getPropertyVal
    ue(description).toString();
41.     }
42.     // 获取对象元模型的形状属性
43.     String metaObjectShape = ontMetaObject.getPropertyValue(shap
    e).toString();
44.     // 获取对象元模型的模式属性
45.     String metaObjectScreenMode = ontMetaObject.getPropertyValue
    (annotationPropertyScreenMode)
46.       .toString();// visualizedMode
47.     // 获取对象元模型的图片地址属性
48.     if (ontMetaObject.getPropertyValue(annotationPropertyImageAd
    dress) != null) {
49.      String metaObjectImageAddress = ontMetaObject.getPropertyVa
    lue(annotationPropertyImageAddress)
50.        .toString();
51.     }
52.     // 获取对象元模型的图标属性
53.     if (ontMetaObject.getPropertyValue(annotationPropertyIcon) !
    = null) {
54.      String metaObjectIcon = ontMetaObject.getPropertyValue(anno
    tationPropertyIcon).toString();
55.     }
56.     // 获取对象元模型的展开方向属性
57.     if (ontMetaObject.getPropertyValue(annotationPropertyUnfoldD
    irection) != null) {
58.      String metaObjectUnfoldDirection = ontMetaObject
59.        .getPropertyValue(annotationPropertyUnfoldDirection).toSt
    ring();
60.     }
61.
62.     for (Iterator object_it = ontMetaObject.listEquivalentClasse
    s(); object_it.hasNext();) {
63.      OntClass c2 = (OntClass) object_it.next();
64.      Restriction r2 = c2.asRestriction();
65.      // 获取对象元模型具有的属性元模型
66.      if (r2.isSomeValuesFromRestriction()
67.        && r2.getOnProperty().getLocalName().equals("hasProperty"
    )) {
68.       SomeValuesFromRestriction sr2 = r2.asSomeValuesFromRestric
    tion();
```

```
69.       OntClass metaObjectProperty = (OntClass) sr2.getSomeValues
   From();
70.       System.out.println(metaObjectProperty);
71.      }
72.      if (r2.isAllValuesFromRestriction()
73.        && r2.getOnProperty().getLocalName().equals("hasProperty"
   )) {
74.       AllValuesFromRestriction sr2 = r2.asAllValuesFromRestricti
   on();
75.       OntClass metaObjectProperty = (OntClass) sr2.getAllValuesF
   rom();
76.      }
77.      // 获取对象元模型具有的点元模型
78.      if (r2.isSomeValuesFromRestriction()
79.        && r2.getOnProperty().getLocalName().equals("linkObjectAn
   dPoint")) {
80.       SomeValuesFromRestriction sr2 = r2.asSomeValuesFromRestric
   tion();
81.       OntClass metaObjectIntersectionClass = (OntClass) sr2.getS
   omeValuesFrom();
82.       // IntersectionClass（交集类）是一种本体构造，用于表示两个或多
   个类的交集
83.       if (metaObjectIntersectionClass.isIntersectionClass()) {
84.        OntClass metaObjectPoint = null;
85.        for (Iterator and = metaObjectIntersectionClass.asInterse
   ctionClass()
86.          .listOperands(); and.hasNext();) {
87.         OntClass and_class = (OntClass) and.next();
88.         if (!and_class.isRestriction()) {
89.          metaObjectPoint = and_class;
90.         } else {
91.          // 获取点的方向(数据属性)
92.          String pointDirection = and_class.asRestriction().asHas
   ValueRestriction()
93.            .getHasValue().toString();
94.         }
95.        }
96.
97.       }
98.      }
99.
100.     }
101.    }
102.   }
```

```
103.  }
104. }
```

7.3 案例

通过调用 getMetaObject 方法获取的信息如下所示：

其中 Object_Block 为 metaObjectName，也就是对象元模型的 id 属性，模块为 metaObjectLocalLabel 也就是对象元模型的 localLabel 属性，其中 Property_Name 为对象原模型的属性元模型，Point_Full_Port 等为对象元模型具有的点元模型。

（1） Protégé工具

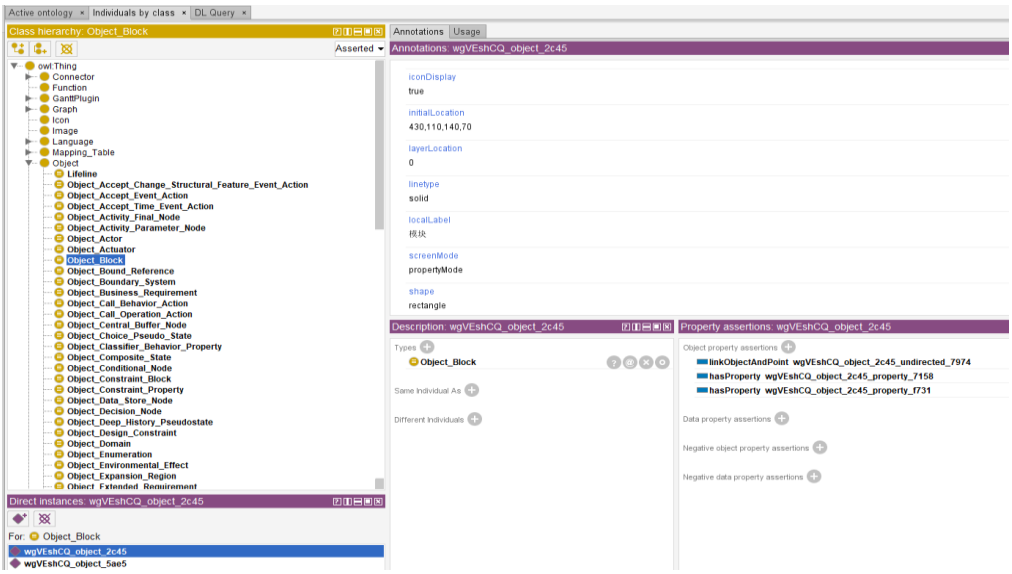

图 7-1对对象元模型信息（本体）

（2） MetaGraph工具

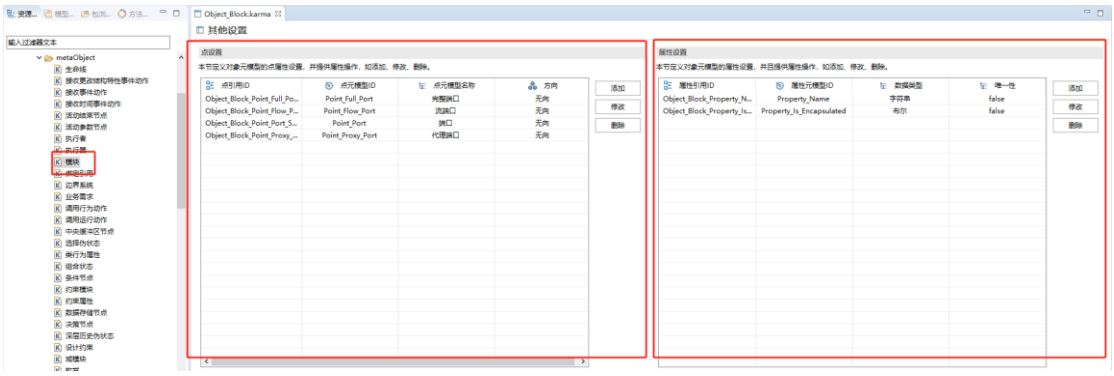

图 7-3对象元模型信息中的点模型以及属性模型信息（MetaGraph）

（3） 对象元模型中的点元模型以及属性元模型

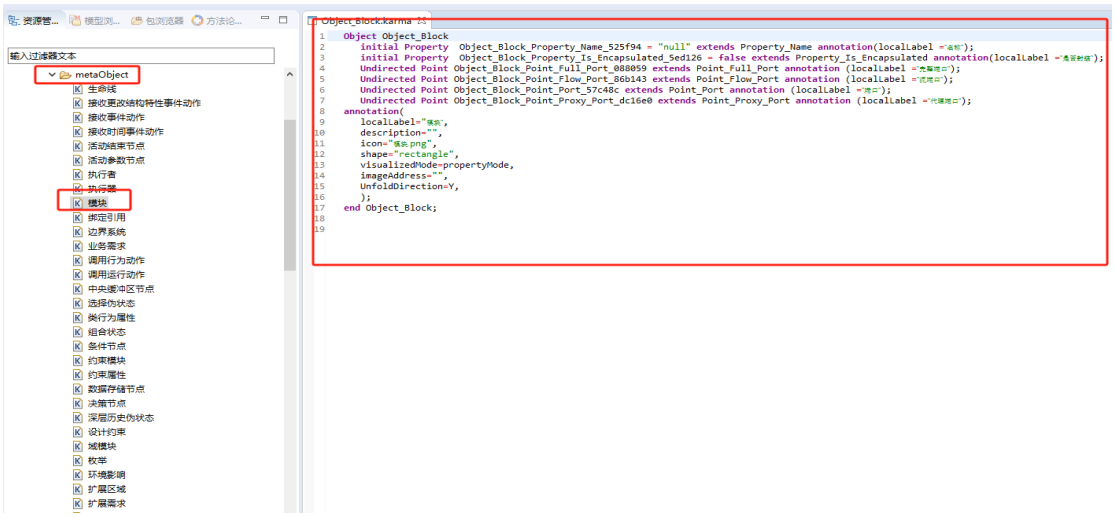

图 7-2对象元模型信息（MetaGraph）

8 获取关系元模型信息

8.1 目的

主要目的是处理本体数据，从指定OWL文件中读取关系元模型信息，具体信息如下：

（1） getMetaRelationship该方法是一个静态公共方法，用于接收一个文件路径path作为参数;

（2） 声明METAG变量用于定义一个命名空间（namespace）的URL，用于在本体中唯一实体标识、属性等元素;

（3） 创建OntModel类型的本体模型对象ontModel, 使用了OntModelSpec.Owl_Men规范,同时将本体模型设置为非严格模式,避免读取实例时报错;

（4） 通过调用read方法，将具体的本体数据加载到ontModel对象中，供后续操作;

（5） 通过调用getAnnotationProperty方法，获取本体注释属性localLabel, description,shape等,描述关系元模型的各种属性；

（6） 通过调用getOntClass方法，获取本体中的Language类，表示MetaGraph中的语言文件夹;

（7） 通过调用getObjectProperty方法，获取本体的对象属性languageIncludingRelationship，该属性表示元素之间的关系，语言包括的关系元模型;

（8） 通过嵌套的循环遍历了 Language 类的子类以及与这些子类等价的类；

（9）在循环中，代码判断等价类是否具有属性languageIncludingRelationship，如果是，则进一步获取该建模语言的关系元模型信息，包括id，locallabel等；

（10）同时在内部循环中，获取关系元模型具有的属性元模型和角色元模

型信息，通过遍历ontMetaRelationship的等价类列表，获取具有关系的hasProperty属性元模型和角色元模型

8.2 代码

```
1.  // 获取关系元模型信息
2.  public static void getMetaRelationship(String path) {
3.    // 命名空间（namespace）的URL，用于在本体建模中为实体、属性和其他元素提供唯一的标识符
4.    String METAG = "http://www.zkhoneycomb.com/formats/metagInOwl#";
5.    // 创建一个OntModel对象
6.    OntModel ontModel = ModelFactory.createOntologyModel(OntModelSpec.OWL_MEM);
7.    // 从指定的文件路径(path)读取本体数据并将其加载到ontModel对象中。
8.    ontModel.read(path);
9.    // 读取本体的注释属性
10.   AnnotationProperty localLabel = ontModel.getAnnotationProperty(METAG + "localLabel");
11.   AnnotationProperty description = ontModel.getAnnotationProperty(METAG + "description");
12.   AnnotationProperty shape = ontModel.getAnnotationProperty(METAG + "shape");
13.   AnnotationProperty annotationPropertyIcon = ontModel.getAnnotationProperty(METAG + "icon");
14.   AnnotationProperty id = ontModel.getAnnotationProperty(METAG + "id");
15.
16.   OntClass language = ontModel.getOntClass(METAG + "Language");
17.
18.   ObjectProperty languageIncludingRelationship = ontModel
19.     .getObjectProperty(METAG + "languageIncludingRelationship");
20.
21.   for (ExtendedIterator<OntClass> it = language.listSubClasses(); it.hasNext();) {
22.     OntClass ontLanguage = it.next();
23.     String ontLanguageId = ontLanguage.getPropertyValue(id).toString();
24.     String ontLanguageName = ontLanguage.getPropertyValue(localLabel).toString();
```

```
25.
26.   for (ExtendedIterator<OntClass> it2 = ontLanguage.listEquivale
   ntClasses(); it2.hasNext();) {
27.    OntClass equivalentClass = it2.next();
28.    if (equivalentClass.asRestriction().getOnProperty().equals(la
   nguageIncludingRelationship)) {
29.     OntClass ontMetaRelationship = (OntClass) equivalentClass.as
   Restriction()
30.       .asSomeValuesFromRestriction().getSomeValuesFrom();
31.     // 获取关系元模型的 ID 属性
32.     String metaRelationshipName = ontMetaRelationship.getLocalNa
   me();
33.     // 获取关系元模型的 localLabel 属性
34.     String metaRelationshipLocalLabel = ontMetaRelationship.getP
   ropertyValue(localLabel).toString();
35.     // 获取关系元模型的 Icon 属性
36.     if (ontMetaRelationship.getPropertyValue(annotationPropertyI
   con) != null) {
37.      String metaRelationshipIcon = ontMetaRelationship.getProper
   tyValue(annotationPropertyIcon)
38.        .toString();
39.     }
40.     // 获取关系元模型的 description 属性
41.     if (ontMetaRelationship.getPropertyValue(description) != nul
   l) {
42.      String metaRelationshipDescription = ontMetaRelationship.ge
   tPropertyValue(description)
43.        .toString();
44.     }
45.     // 获取关系元模型的形状属性
46.     String metaRelationshipShape = ontMetaRelationship.getProper
   tyValue(shape).toString();
47.
48.     // 获取关系元模型具有的属性元模型和角色元模型
49.     for (Iterator relationship_it = ontMetaRelationship.listEqui
   valentClasses(); relationship_it
50.       .hasNext();) {
51.      OntClass c2 = (OntClass) relationship_it.next();
52.      Restriction r2 = c2.asRestriction();
53.      // 获取关系元模型具有的属性元模型
54.      if (r2.isSomeValuesFromRestriction()
55.        && r2.getOnProperty().getLocalName().equals("hasProperty"
   )) {
```

```
56.       SomeValuesFromRestriction sr2 = r2.asSomeValuesFromRestric
   tion();
57.       OntClass metaRelationshipProperty = (OntClass) sr2.getSome
   ValuesFrom();
58.      }
59.      if (r2.isAllValuesFromRestriction()
60.        && r2.getOnProperty().getLocalName().equals("hasProperty"
   )) {
61.       AllValuesFromRestriction sr2 = r2.asAllValuesFromRestricti
   on();
62.       OntClass metaRelationshipProperty = (OntClass) sr2.getAllV
   aluesFrom();
63.      }
64.      // 获取关系元模型具有的角色元模型
65.      if (r2.isSomeValuesFromRestriction()
66.        && r2.getOnProperty().getLocalName().equals("linkRelation
   shipAndRole")) {
67.       SomeValuesFromRestriction sr2 = r2.asSomeValuesFromRestric
   tion();
68.       OntClass metaRelationshipRole = (OntClass) sr2.getSomeValu
   esFrom();
69.      }
70.     }
71.    }
72.   }
73.
74.  }
75.
76. }
```

8.3 案例

通过调用 getMetaRelationship 方法获取的信息如下所示：

其中 Message 为 metaRelationshipName 也就是关系元模型的 id 属性，消息为 metaRelationshipLocalLabel 也就是关系元模型的 locallabel 属性，其中 Property_On_Port 等为关系元模型的属性元模型，Message_Source 等为关系元模型的角色元模型。

（1） Protégé工具

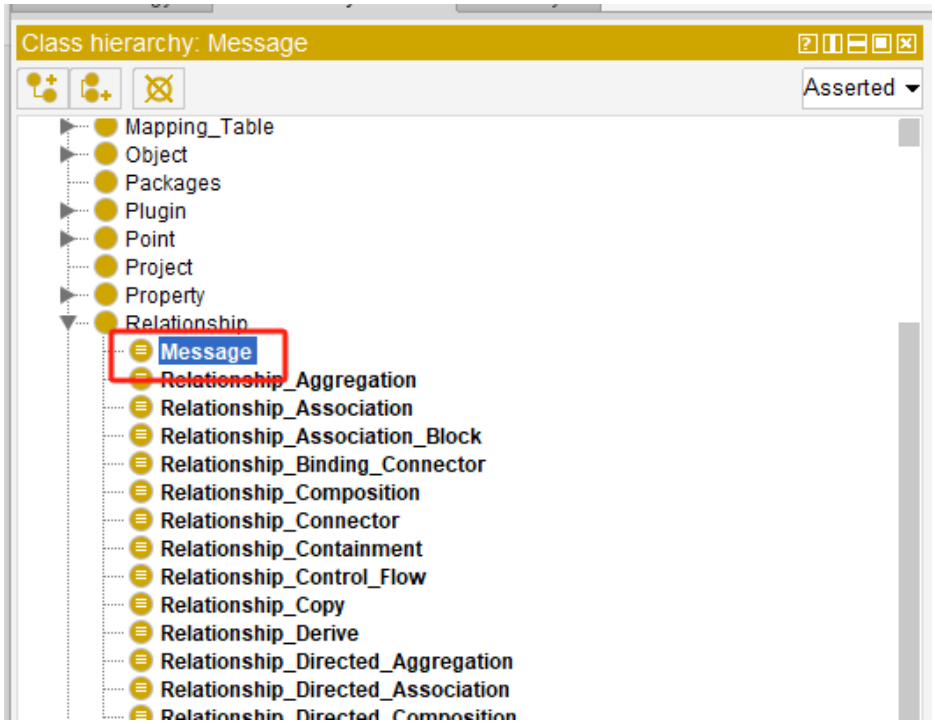

图 8-1关系元模型信息（本体）

（2） MetaGraph工具

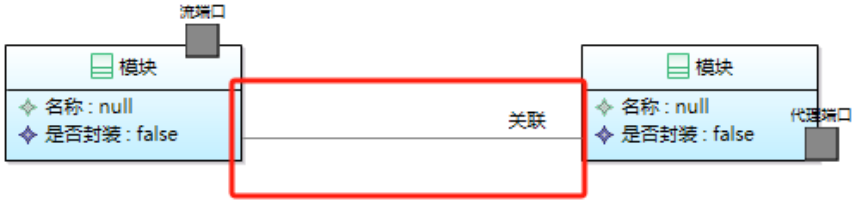

图 8-2关系元模型信息（MetaGraph）

（3） 关系元模型中的属性元模型

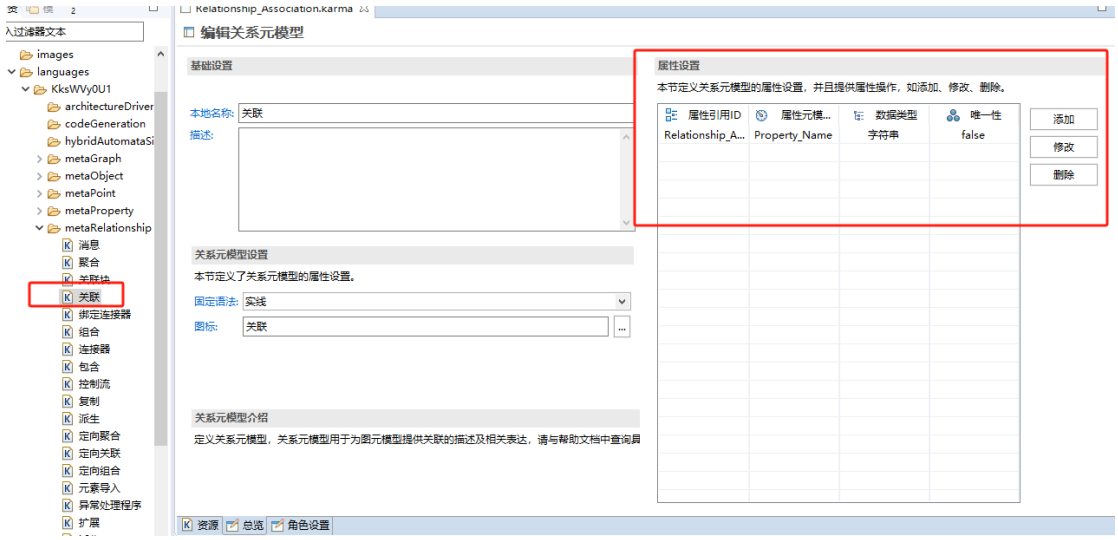

图 8-3关系元模型的属性元模型（MetaGraph）

（4） 关系元模型中的角色元模型

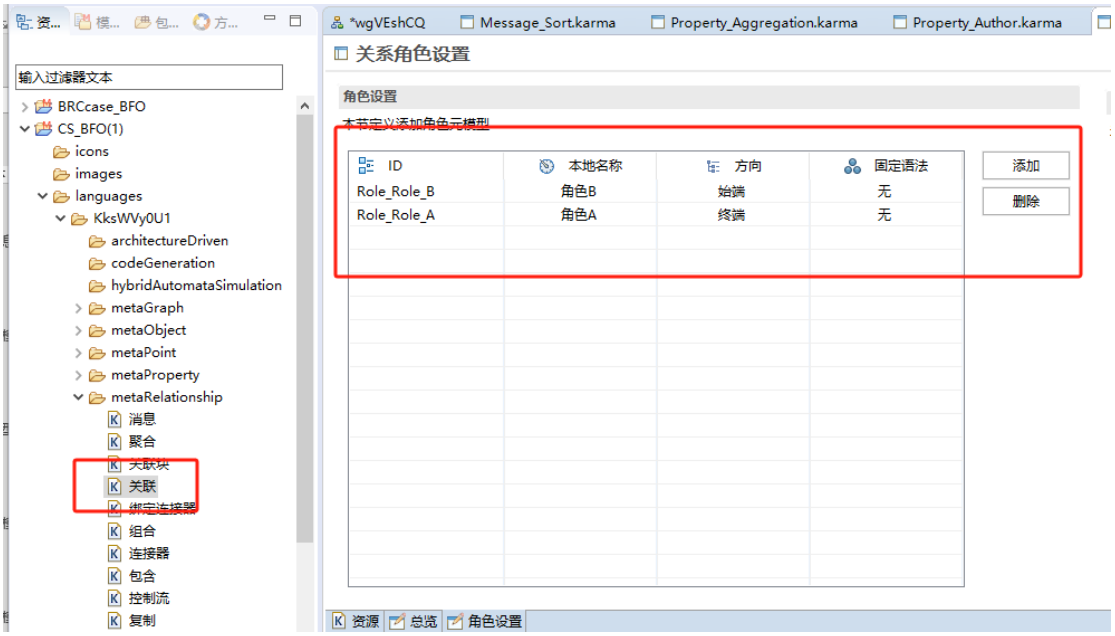

图 8-4关系元模型的角色元模型（MetaGraph）

9 获取图元模型信息

9.1 目的

主要目的是处理本体数据 ,从指定OWL文件中读取图元模型信息 ,具体信息

如下：

（1） getMetaGraph该方法是一个静态公共方法，用于接收一个文件路径path作为参数;

（2） 声明METAG变量用于定义一个命名空间（namespace）的URL，用于在本体中唯一实体标识、属性等元素;

（3） 创建OntModel类型的本体模型对象ontModel, 使用了OntModelSpec.Owl_Men规范,同时将本体模型设置为非严格模式,避免读取实例时报错;

（4） 通过调用read方法，将具体的本体数据加载到ontModel对象中，供后续操作;

（5） 通过调用getAnnotationProperty方法，获取本体注释属性localLabel, description,shape等,描述图元模型的各种属性；

（6） 通过调用getOntClass方法，获取本体中的Language类，表示MetaGraph中的语言文件夹;

（7） 通过调用getObjectProperty方法，获取本体的对象属性languageIncludingGraph，该属性表示元素之间的关系，语言包括的图元模型;

（8） 通过嵌套的循环遍历了 Language类的子类以及与这些子类等价的类；

（9） 在循环中，代码判断等价类是否具有属性languageIncludingGraph，如果是，则进一步获取该建模语言的图元模型信息，包括id，locallabel等；

（10）同时在内部循环中，获取图元模型具有的属性元模型、对象元模型、元关系模型和约束信息；

（11）获取图元模型中分割和剖视的信息，包括分割分割视的对象元模型、图元模型以及分割/剖视的类型；

9.2 代码

```
1.  // 获取图元模型信息
2.  public static void getMetaGraph(String path) {
3.    // 命名空间（namespace）的 URL，用于在本体建模中为实体、属性和其他元素提供唯一的标识符
4.    String METAG = "http://www.zkhoneycomb.com/formats/metagInOwl#";
5.    // 创建一个 OntModel 对象
6.    OntModel ontModel = ModelFactory.createOntologyModel(OntModelSpec.OWL_MEM);
7.    // 从指定的文件路径(path)读取本体数据并将其加载到 ontModel 对象中。
8.    ontModel.read(path);
9.    // 读取本体的注释属性
10.   AnnotationProperty localLabel = ontModel.getAnnotationProperty(METAG + "localLabel");
11.   AnnotationProperty description = ontModel.getAnnotationProperty(METAG + "description");
12.   AnnotationProperty annotationPropertyType = ontModel.getAnnotationProperty(METAG + "type");
13.   AnnotationProperty id = ontModel.getAnnotationProperty(METAG + "id");
14.
15.   OntClass language = ontModel.getOntClass(METAG + "Language");
16.   ObjectProperty languageIncludingGraph = ontModel.getObjectProperty(METAG + "languageIncludingGraph");
17.
18.   for (ExtendedIterator<OntClass> it = language.listSubClasses(); it.hasNext();) {
19.     OntClass ontLanguage = it.next();
20.     String ontLanguageId = ontLanguage.getPropertyValue(id).toString();
21.     String ontLanguageName = ontLanguage.getPropertyValue(localLabel).toString();
22.
23.     for (ExtendedIterator<OntClass> it2 = ontLanguage.listEquivalentClasses(); it2.hasNext();) {
```

```
24.    OntClass equivalentClass = it2.next();
25.    if (equivalentClass.asRestriction().getOnProperty().equals(la
   nguageIncludingGraph)) {
26.     OntClass ontMetaGraph = (OntClass) equivalentClass.asRestric
   tion().asSomeValuesFromRestriction()
27.       .getSomeValuesFrom();
28.     // 获取图元模型的 ID 属性
29.     String metaGraphName = ontMetaGraph.getLocalName();
30.     // 获取图元模型的 localLabel 属性
31.     String metaGraphLocalLabel = ontMetaGraph.getPropertyValue(l
   ocalLabel).toString();
32.     // 获取图元模型的类型属性
33.     if (ontMetaGraph.hasProperty(annotationPropertyType)) {
34.      String metaGraphType = ontMetaGraph.getPropertyValue(annota
   tionPropertyType).toString();
35.     }
36.     // 获取图元模型的描述属性
37.     if (ontMetaGraph.getPropertyValue(description) != null) {
38.      String metaGraphDescription = ontMetaGraph.getPropertyValue
   (description).toString();
39.     }
40.
41.     OntClass metaGraphConnector = null;
42.     for (Iterator graph_it = ontMetaGraph.listEquivalentClasses(
   ); graph_it.hasNext();) {
43.      OntClass c2 = (OntClass) graph_it.next();
44.      Restriction r2 = c2.asRestriction();
45.      // 获取图元模型具有的属性元模型
46.      if (r2.isSomeValuesFromRestriction()
47.        && r2.getOnProperty().getLocalName().equals("hasProperty"
   )) {
48.       SomeValuesFromRestriction sr2 = r2.asSomeValuesFromRestric
   tion();
49.       OntClass metaGraphProperty = (OntClass) sr2.getSomeValuesF
   rom();
50.      }
51.      if (r2.isAllValuesFromRestriction()
52.        && r2.getOnProperty().getLocalName().equals("hasProperty"
   )) {
53.       AllValuesFromRestriction sr2 = r2.asAllValuesFromRestricti
   on();
54.       OntClass metaGraphProperty = (OntClass) sr2.getAllValuesFr
   om();
55.      }
```

```
56.       // 获取图元模型具有的对象元模型
57.       if (r2.isSomeValuesFromRestriction()
58.         && r2.getOnProperty().getLocalName().equals("graphIncludi
   ngObject")) {
59.        SomeValuesFromRestriction sr2 = r2.asSomeValuesFromRestric
   tion();
60.        OntClass metaGraphObject = (OntClass) sr2.getSomeValuesFro
   m();
61.        System.out.println(metaGraphObject);
62.       }
63.       // 获取图元模型具有的关系元模型
64.       if (r2.isSomeValuesFromRestriction()
65.         && r2.getOnProperty().getLocalName().equals("graphIncludi
   ngRelationship")) {
66.        SomeValuesFromRestriction sr2 = r2.asSomeValuesFromRestric
   tion();
67.        OntClass metaGraphRelationship = (OntClass) sr2.getSomeVal
   uesFrom();
68.        System.out.println(metaGraphRelationship);
69.       }
70.       // 获取图元模型具有的约束
71.       if (r2.isSomeValuesFromRestriction()
72.         && r2.getOnProperty().getLocalName().equals("graphIncludi
   ngConnector")) {
73.        SomeValuesFromRestriction sr2 = r2.asSomeValuesFromRestric
   tion();
74.        metaGraphConnector = (OntClass) sr2.getSomeValuesFrom();
75.        System.out.println(metaGraphConnector);
76.       }
77.      }
78.
79.      // 图元模型 Constraint
80.      if (metaGraphConnector != null) {
81.       for (Iterator connector_i = metaGraphConnector.listEquivale
   ntClasses(); connector_i
82.         .hasNext();) {
83.        OntClass c3 = (OntClass) connector_i.next();
84.        // 获取图元模型的约束具体内容
85.        if (c3.isIntersectionClass()) {
86.         for (Iterator and_i = c3.asIntersectionClass().listOperan
   ds(); and_i.hasNext();) {
87.          OntClass and_one = (OntClass) and_i.next();
88.          if (and_one.isRestriction()) {
```

```
89.          OntClass c4 = (OntClass) and_one.asRestriction().asSome
   ValuesFromRestriction()
90.            .getSomeValuesFrom();
91.          // 获取图元模型中角色元模型的约束
92.          if (c4.getSuperClass().getLocalName().equals("Role")) {
93.           String connector_role = c4.getLocalName();
94.           System.out.println(connector_role);
95.          }
96.          // 获取图元模型中对象元模型的约束
97.          if (c4.getSuperClass().getLocalName().equals("Object"))
    {
98.           String connector_object = c4.getLocalName();
99.           System.out.println(connector_object);
100.          }
101.          // 获取图元模型中点元模型的约束
102.          if (c4.getSuperClass().hasSuperClass()
103.            && c4.getSuperClass().getSuperClass().getLocalName()
   .equals("Point")) {
104.           String connector_point = c4.getLocalName();
105.           System.out.println(connector_point);
106.          }
107.          // 获取图元模型中关系元模型的约束
108.          if (c4.getSuperClass().getLocalName().equals("Relation
   ship")) {
109.           String connector_relationship = c4.getLocalName();
110.           System.out.println(connector_relationship);
111.          }
112.
113.         }
114.        }
115.        System.out.println("one Connector over");
116.       }
117.       // 获取图元模型中具有分解剖视的对象元模型，以及对象元模型分解剖
   视的图元模型
118.       if (c3.isRestriction()) {
119.        Restriction c4 = c3.asRestriction();
120.        String DorE = null;// 记录是分解还是剖视(D or E)
121.        String DorE_object = null;// 记录分解剖视的对象
122.        String DorE_graph = null;// 记录分解剖视的图
123.        if (((OntClass) c4.asSomeValuesFromRestriction().getSome
   ValuesFrom())
124.          .isIntersectionClass()) {
125.         for (Iterator c4_i = ((OntClass) c4.asSomeValuesFromRes
   triction()
```

```
            .getSomeValuesFrom()).asIntersectionClass().listOperands(); c4_i
              .hasNext();) {
          OntClass c4_i_c = (OntClass) c4_i.next();
          if (c4_i_c.isRestriction()) {
           Restriction c4_i_r = c4_i_c.asRestriction();
           if (c4_i_r.isSomeValuesFromRestriction()) {
            DorE = c4_i_r.getOnProperty().getLocalName();
            // 获取对象元模型分解剖视的图元模型
            DorE_graph = c4_i_r.asSomeValuesFromRestriction().getSomeValuesFrom()
               .getLocalName();
           }

          } else {
           // 获取具有分解剖视的对象元模型
           DorE_object = c4_i_c.getLocalName();
          }
         }
        }
       }

     }

    }
   }
  }
 }
```

9.3 案例

通过调用 getMetaGraph 方法获取的信息如下所示：

其中 Block_Defintion_Diagram 为 metaGraphName 即为图元模型的 id 属性，模块定义图为 metaGraphLocalLabel 即为图元模型的 locallabel 属性，其中 Property_Context 等为图元模型的的属性元模型，Object_Block 等为图元模型的对象元模型，Relationship_Association 等为图元模型的关系元模型，

（1） Protégé工具

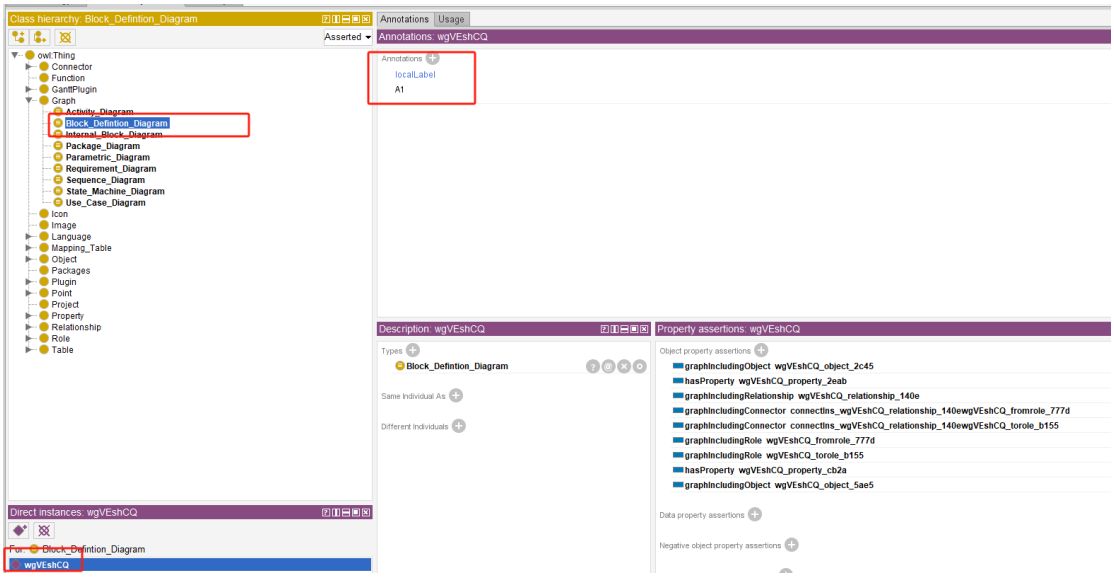

图 9-1图元模型信息（本体）

（2） MetaGraph工具

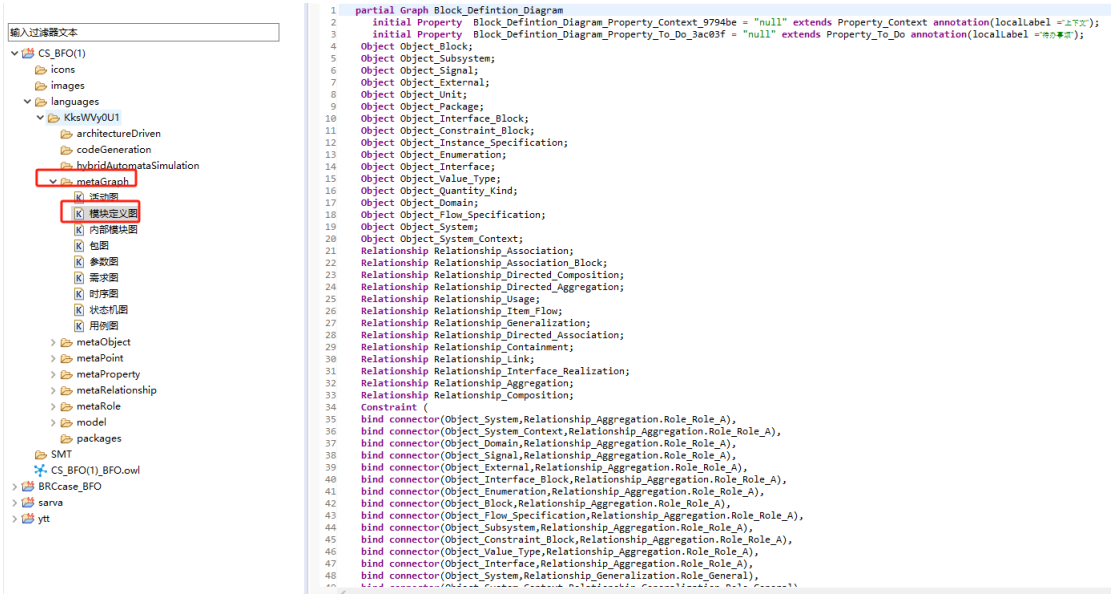

图 9-2图元模型信息（MetaGraph）

（3） 下图左侧为图元模型中的对象元模型，右侧为图元模型中的关系元

模型

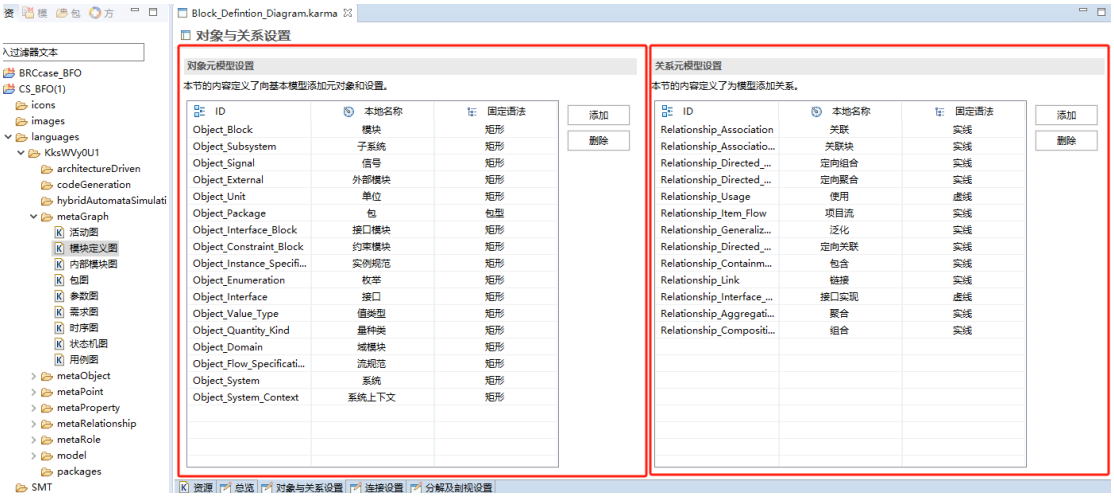

图 10-1图元模型信息中的对象元模型以及关系元模型（MetaGraph）

10 获取模型信息

10.1 目的

主要目的是处理本体数据,从指定OWL文件中读取模型信息,获取模型中的对象，关系以及角色等，具体信息如下：

（1） getModel该方法是一个静态公共方法,用于接收一个文件路径path作为参数;

（2） 声明METAG变量用于定义一个命名空间(namespace)的URL,用于在本体中唯一实体标识、属性等元素;

（3） 创建OntModel类型的本体模型对象ontModel,使用了OntModelSpec.Owl_Men规范,同时将本体模型设置为非严格模式,避免读取实例时报错;

（4） 通过调用read方法,将具体的本体数据加载到ontModel对象中,供后续操作;

（5） 通过调用getAnnotationProperty方法，获取本体注释属性localLabel, description,shape等,描述模型的各种属性；然后，通过命名空间，创建了一系列AnnotationProperty对象，获取用于特定注释属性的信息。这些注释属性包括模型的初始位置、本地标签、形状、颜色等；

（6） 通过调用getOntClass方法，获取本体中的Language类，表示MetaGraph中的语言文件夹;

（7） 通过调用getObjectProperty方法，获取本体的对象属性languageIncludingGraph，该属性表示元素之间的关系，语言包括的模型;

（8） 通过嵌套的循环遍历了 Language 类的子类以及与这些子类等价的类；

（9） 在循环中，代码判断等价类是否具有属性languageIncludingGraph，如果是，则进一步获取该建模语言的模型信息，包括id，locallabel等；

（10）循环具体模型，获取属性谓词的localname,判断是否和hasProperty相等，如果是获取模型的图属性和图属性值；

（11）循环具体模型，获取属性谓词的localname,判断是否和graphIncludingObject相等，如果是获取模型的对象实例，对象实例的属性实例和属性值，对象实例的注释属性，

（12）循环具体模型，获取属性谓词的localname,判断是否分别和explode，decompose，cloneModel，cloneObject相等，如果想等则分别获取获取模型的对象实例剖视、分解、克隆模型、克隆对象的实例；

（13）获取模型的对象实例的点实例，及其具有的属性实例和属性值；

（14）循环具体模型，获取属性谓词的localname,判断是否和

graphIncludingRelationship相等，如果是获取模型的关系实例，关系实例的属性实例和属性值，关系实例的注释属性，

（15）循环具体的关系实例的所有属性，获取关系实例的角色实例及其属性实例；

（16）同时获取关系实例的属性实例和属性值；

（17）循环具体模型，获取属性谓词的localname，判断是否和graphIncludingConnector相等，如果是获取模型的约束实例，循环约束实例的所有属性，判断属性的谓词的localname是否和分别和linkFromRelationship，linkRelationshipAndRole，linkToObject，linkObjectAndPoint相等，如果相等则分别获取约束关系实例，约束关系和角色实例，约束对象实例，约束对象和点实例以及约束方式；

10.2 代码

```
1.  // 获取模型信息
2.  public static void getModel(String path) {
3.   // 命名空间（namespace）的URL，用于在本体建模中为实体、属性和其他元素提供唯一的标识符
4.   String METAG = "http://www.zkhoneycomb.com/formats/metagInOwl#";
5.   // 创建一个OntModel对象
6.   OntModel ontModel = ModelFactory.createOntologyModel(OntModelSpec.OWL_MEM);
7.   // 从指定的文件路径(path)读取本体数据并将其加载到ontModel对象中。
8.   ontModel.read(path);
9.   // 读取本体的注释属性
10.  AnnotationProperty initialLocation = ontModel.getAnnotationProperty(METAG + "initialLocation");
11.  AnnotationProperty localLabel = ontModel.getAnnotationProperty(METAG + "localLabel");
12.  AnnotationProperty shape = ontModel.getAnnotationProperty(METAG + "shape");
```

```
13.  AnnotationProperty annotationPropertyColor = ontModel.createAnn
   otationProperty(METAG + "color");
14.  AnnotationProperty screenMode = ontModel.getAnnotationProperty(
   METAG + "screenMode");
15.  AnnotationProperty imageAddress = ontModel.getAnnotationPropert
   y(METAG + "imageAddress");
16.  AnnotationProperty annotationPropertySdIdentification = ontMode
   l
17.     .getAnnotationProperty(METAG + "SdIdentification");
18.  AnnotationProperty annotationPropertyText = ontModel.createAnno
   tationProperty(METAG + "text");
19.  AnnotationProperty annotationPropertyIconDisplay = ontModel.cre
   ateAnnotationProperty(METAG + "iconDisplay");
20.  AnnotationProperty annotationPropertyFrameSize = ontModel.getAn
   notationProperty(METAG + "frameSize");
21.  AnnotationProperty annotationPropertyLineType = ontModel.getAnn
   otationProperty(METAG + "linetype");
22.
23.  AnnotationProperty lableDisplay = ontModel.getAnnotationPropert
   y(METAG + "lableDisplay");
24.
25.  AnnotationProperty annotationPropertyPolyLineLocation = ontMode
   l
26.     .createAnnotationProperty(METAG + "polyLineLocation");
27.  AnnotationProperty startLocation = ontModel.createAnnotationPro
   perty(METAG + "startLocation");
28.  AnnotationProperty endLocation = ontModel.createAnnotationPrope
   rty(METAG + "endLocation");
29.  AnnotationProperty annotationPropertyCifString = ontModel.creat
   eAnnotationProperty(METAG + "cifString");
30.  AnnotationProperty annotationPropertyStyle = ontModel.createAnn
   otationProperty(METAG + "style");
31.  AnnotationProperty annotationPropertyInitial = ontModel.createA
   nnotationProperty(METAG + "initial");
32.  AnnotationProperty annotationPropertyType = ontModel.createAnno
   tationProperty(METAG + "type");
33.  AnnotationProperty annotationPropertyModelSize = ontModel.creat
   eAnnotationProperty(METAG + "modelSize");
34.  AnnotationProperty annotationPropertySmooth = ontModel.createAn
   notationProperty(METAG + "smooth");
35.  AnnotationProperty annotationPropertyJumpStatus = ontModel.crea
   teAnnotationProperty(METAG + "jumpStatus");
36.  AnnotationProperty annotationPropertyJumpType = ontModel.create
   AnnotationProperty(METAG + "jumpType");
```

```
37.  AnnotationProperty annotationPropertyJumpReverse = ontModel.cre
   ateAnnotationProperty(METAG + "jumpReverse");
38.  AnnotationProperty annotationPropertyAvoidObstructions = ontMod
   el
39.     .createAnnotationProperty(METAG + "avoidObstructions");
40.  AnnotationProperty annotationPropertyClosestDistance = ontModel
41.     .createAnnotationProperty(METAG + "closestDistance");
42.  AnnotationProperty annotationPropertyQueue = ontModel.createAnn
   otationProperty(METAG + "queue");
43.  AnnotationProperty annotationPropertyRoutingType = ontModel.get
   AnnotationProperty(METAG + "routingType");
44.  AnnotationProperty annotationPropertyStrikethrough = ontModel.g
   etAnnotationProperty(METAG + "strikeThrough");
45.  AnnotationProperty annotationPropertyUnderline = ontModel.getAn
   notationProperty(METAG + "underline");
46.
47.  ObjectProperty decomposeProp = ontModel.createObjectProperty(ME
   TAG + "decompose");
48.  ObjectProperty explodeProp = ontModel.createObjectProperty(META
   G + "explode");
49.  ObjectProperty hasPropertyProp = ontModel.createObjectProperty(
   METAG + "hasProperty");
50.
51.  OntClass language = ontModel.getOntClass(METAG + "Language");
52.
53.  ObjectProperty languageIncludingGraph = ontModel.getObjectPrope
   rty(METAG + "languageIncludingGraph");
54.  for (ExtendedIterator<OntClass> it = language.listSubClasses();
    it.hasNext();) {
55.   OntClass ontLanguage = it.next();
56.   String ontLanguageName = ontLanguage.getLocalName();
57.
58.   for (ExtendedIterator<OntClass> it2 = ontLanguage.listEquivale
   ntClasses(); it2.hasNext();) {
59.    OntClass equivalentClass = it2.next();
60.    if (equivalentClass.asRestriction().getOnProperty().equals(la
   nguageIncludingGraph)) {
61.     OntClass ontMetaGraph = (OntClass) equivalentClass.asRestric
   tion().asSomeValuesFromRestriction()
62.       .getSomeValuesFrom();
63.     for (Iterator il2 = ontMetaGraph.listInstances(); il2.hasNex
   t();) {
64.      Individual karmaModel = (Individual) il2.next();
65.      // 获取模型Id属性
```

```
66.       String karmaModelName = karmaModel.getLocalName();
67.       // 获取模型 localLabel 属性
68.       String karmaModelLocalLabel = karmaModel.getPropertyValue(l
   ocalLabel).toString();
69.       // 获取模型 Cif 属性
70.       if (karmaModel.hasProperty(annotationPropertyCifString)) {
71.        String karmaModelCif = karmaModel.getPropertyValue(annotat
   ionPropertyCifString).toString();
72.       }
73.
74.       for (Iterator ig = karmaModel.listProperties(); ig.hasNext(
   );) {
75.        Statement igs = (Statement) ig.next();// Statement 是 Apac
   he Jena 中的类，用于表示 RDF
76.                    // 三元组中的语句，包括主语、谓词和宾语。
77.        // 1.获取模型的属性及属性值
78.        if (igs.getPredicate().getLocalName().equals("hasProperty"
   )) {// igs.getPredicate()获取该属性的谓词
79.         // 获取模型的属性实例
80.         Individual graphProperty = ontModel.getIndividual(igs.get
   Resource().getURI());
81.         // 获取模型的属性值
82.         String graphPropertyValue = "";
83.         for (Iterator igp = graphProperty.listProperties(); igp.h
   asNext();) {
84.          Statement igps = (Statement) igp.next();
85.          if (igps.getPredicate().getLocalName().equals("value"))
   {
86.           graphPropertyValue = igps.getString();
87.          }
88.         }
89.        }
90.        // 2.获取模型的对象实例
91.        if (igs.getPredicate().getLocalName().equals("graphIncludi
   ngObject")) {
92.         // 获取模型的对象实例
93.         Individual graph_Object_Individual = ontModel.getIndividu
   al(igs.getResource().getURI());
94.         String graph_Object_IndividualName = graph_Object_Individ
   ual.getLocalName();
95.         String graph_Object_IndividualLocalLabel = graph_Object_I
   ndividual
96.           .getPropertyValue(localLabel).toString();
```

```
97.         String graph_Object_IndividualShape = graph_Object_Indivi
    dual.getPropertyValue(shape)
98.           .toString();
99.         // 获取模型的对象实例具有的注释属性
100.         if (graph_Object_Individual.hasProperty(annotationProper
    tyColor)) {
101.          String graph_Object_IndividualColor = graph_Object_Indi
    vidual
102.            .getPropertyValue(annotationPropertyColor).toString()
    ;
103.         }
104.         String[] locations = graph_Object_Individual.getProperty
    Value(initialLocation)
105.           .toString().split(",");
106.
107.         if (graph_Object_Individual.hasProperty(annotationProper
    tyCifString)) {
108.          String graph_Object_IndividualCifString = graph_Object_
    Individual
109.            .getPropertyValue(annotationPropertyCifString).toStri
    ng()
110.            .replace("\\\"", "\"");
111.         }
112.         if (graph_Object_Individual.hasProperty(annotationProper
    tyStyle)) {
113.          String graph_Object_IndividualStyle = graph_Object_Indi
    vidual
114.            .getPropertyValue(annotationPropertyStyle).toString()
    ;
115.         }
116.         if (graph_Object_Individual.hasProperty(annotationProper
    tyInitial)) {
117.          String graph_Object_IndividualInitial = graph_Object_In
    dividual
118.            .getPropertyValue(annotationPropertyInitial).toString
    ();
119.         }
120.         if (graph_Object_Individual.hasProperty(annotationProper
    tyType)) {
121.          String graph_Object_IndividualType = graph_Object_Indiv
    idual
122.            .getPropertyValue(annotationPropertyType).toString();
123.         }
124.         if (graph_Object_Individual.hasProperty(imageAddress)) {
```

```
125.          String graph_Object_IndividualImageAddress = graph_Obje
   ct_Individual
126.              .getPropertyValue(imageAddress).toString();
127.          }
128.          if (graph_Object_Individual.hasProperty(annotationProper
   tyIconDisplay)) {
129.          String graph_Object_IndividualIconDisplay = graph_Objec
   t_Individual
130.              .getPropertyValue(annotationPropertyIconDisplay).toSt
   ring();
131.          }
132.          if (graph_Object_Individual.hasProperty(annotationProper
   tyText)) {
133.          String[] texts = graph_Object_Individual.getPropertyVal
   ue(annotationPropertyText)
134.              .toString().split(",");
135.          }
136.          if (graph_Object_Individual.hasProperty(annotationProper
   tyLineType)) {
137.          String graph_Object_IndividualLineType = graph_Object_I
   ndividual
138.              .getPropertyValue(annotationPropertyLineType).toStrin
   g();
139.          }
140.          if (graph_Object_Individual.hasProperty(annotationProper
   tyFrameSize)) {
141.          Integer graph_Object_IndividualFrameSize = Integer.pars
   eInt(graph_Object_Individual
142.              .getPropertyValue(annotationPropertyFrameSize).toStri
   ng());
143.          }
144.          if (graph_Object_Individual.hasProperty(annotationProper
   tySdIdentification)) {
145.          String graph_Object_IndividualSdIdentification = graph_
   Object_Individual
146.              .getPropertyValue(annotationPropertySdIdentification)
   .toString();
147.          }
148.          // 2.1 获取模型的对象实例具有的属性实例及属性值
149.          for (Iterator igp = graph_Object_Individual.listProperti
   es(); igp.hasNext();) {
150.          Statement igps = (Statement) igp.next();
151.          if (igps.getPredicate().getLocalName().equals("hasPrope
   rty")) {
```

```
152.           // 获取模型的对象实例具有的属性实例
153.           Individual object_Property_Individual = ontModel
154.             .getIndividual(igps.getResource().getURI());
155.           // 找到属性值
156.           String object_Property_Individual_Value = "";
157.           for (Iterator igp2 = object_Property_Individual.listPr
    operties(); igp2
158.             .hasNext();) {
159.            Statement igps2 = (Statement) igp2.next();
160.            if (igps2.getPredicate().getLocalName().equals("value
    ")) {
161.             object_Property_Individual_Value = igps2.getString()
    ;
162.            }
163.           }
164.
165.           if (object_Property_Individual.hasProperty(annotationP
    ropertyText)) {
166.            String[] texts = object_Property_Individual
167.              .getPropertyValue(annotationPropertyText).toString(
    ).split(",");
168.           }
169.
170.           if (object_Property_Individual.hasProperty(annotationP
    ropertyColor)) {
171.            String object_Property_Individual_Color = object_Prop
    erty_Individual
172.              .getPropertyValue(annotationPropertyColor).toString
    ();
173.           }
174.           if (object_Property_Individual
175.             .getPropertyValue(annotationPropertyStrikethrough) !
    = null) {
176.            String object_Property_Individual_Strikethrough = obj
    ect_Property_Individual
177.              .getPropertyValue(annotationPropertyStrikethrough).
    toString();
178.           }
179.           if (object_Property_Individual
180.             .getPropertyValue(annotationPropertyUnderline) != nu
    ll) {
181.            String object_Property_Individual_Underline = object_
    Property_Individual
```

```
182.             .getPropertyValue(annotationPropertyUnderline).toSt
   ring();
183.          }
184.
185.          for (Iterator igp2 = object_Property_Individual.listPr
   operties(); igp2
186.            .hasNext();) {
187.           Statement igps2 = (Statement) igp2.next();
188.           if (igps2.getPredicate().getLocalName().equals("clone
   Property")) {
189.            Individual object_property_clone_Individual = ontMod
   el
190.              .getIndividual(igps2.getResource().getURI());
191.            String object_property_clone = object_property_clone
   _Individual
192.              .getLocalName();
193.           }
194.          }
195.
196.         }
197.         // 2.2 获取模型的对象实例剖视、分解、克隆模型、克隆对象的实
   例
198.         if (igps.getPredicate().getLocalName().equals("explode"
   )) {
199.          Individual explodeIndi = ontModel.getIndividual(igps.g
   etResource().getURI());
200.         }
201.         if (igps.getPredicate().getLocalName().equals("decompos
   e")) {
202.          Individual decomposeIndi = ontModel.getIndividual(igps
   .getResource().getURI());
203.         }
204.         if (igps.getPredicate().getLocalName().equals("cloneMod
   el")) {
205.          Individual cloneModelIndi = ontModel.getIndividual(igp
   s.getResource().getURI());
206.         }
207.         if (igps.getPredicate().getLocalName().equals("cloneObj
   ect")) {
208.          Individual clonObjectIndi = ontModel.getIndividual(igp
   s.getResource().getURI());
209.         }
210.         // 2.3 获取模型的对象实例具有的点实例
```

```
211.          if (igps.getPredicate().getLocalName().equals("linkObje
   ctAndPoint")) {
212.           Individual point_Individual = ontModel
213.             .getIndividual(igps.getResource().getURI());
214.           String point_Individual_Id = point_Individual.getLocal
   Name();
215.           String point_Individual_LocalLabel = point_Individual
216.             .getPropertyValue(localLabel).toString();
217.           String point_Individual_Shape = point_Individual.getPr
   opertyValue(shape)
218.             .toString();
219.           String point_Individual_Color = point_Individual
220.             .getPropertyValue(annotationPropertyColor).toString(
   );
221.           String[] locations2 = point_Individual.getPropertyValu
   e(initialLocation)
222.             .toString().split(",");
223.           String point_Individual_LableDisplay = point_Individua
   l
224.             .getPropertyValue(lableDisplay).toString();
225.           if (point_Individual.hasProperty(annotationPropertyTex
   t)) {
226.            String[] texts = point_Individual.getPropertyValue(an
   notationPropertyText)
227.               .toString().split(",");
228.           }
229.
230.           // 2.3.1 获取点实例的属性实例
231.           for (Iterator igp2 = point_Individual.listProperties()
   ; igp2.hasNext();) {
232.            Statement igps2 = (Statement) igp2.next();
233.            if (igps2.getPredicate().getLocalName().equals("hasPr
   operty")) {
234.             // 找到点实例属性
235.             Individual point_Property_Individual = ontModel
236.               .getIndividual(igps2.getResource().getURI());
237.             for (Iterator igp3 = point_Property_Individual.listP
   roperties(); igp3
238.               .hasNext();) {
239.              Statement igps3 = (Statement) igp3.next();
240.              if (igps3.getPredicate().getLocalName().equals("val
   ue")) {
241.               String point_Property_Individual_Value = igps3.get
   String();
```

```
242.              }
243.             }
244.            }
245.            if (igps2.getPredicate().getLocalName().equals("clone
    Point")) {
246.             Individual clonePointIndi = ontModel
247.               .getIndividual(igps2.getResource().getURI());
248.            }
249.
250.           }
251.          }
252.
253.         }
254.        }
255.        // 3. 获取模型的关系实例及对应的信息
256.        if (igs.getPredicate().getLocalName().equals("graphInclud
    ingRelationship")) {
257.         // 获取关系实例
258.         Individual graph_Rela_Individual = ontModel.getIndividua
    l(igs.getResource().getURI());
259.
260.         // 3.1 获取关系实例两端的角色实例及相关信息
261.         for (Iterator igp = graph_Rela_Individual.listProperties
    (); igp.hasNext();) {
262.          Statement igps = (Statement) igp.next();
263.          // 3.1.1 获取关系实例两端的角色实例
264.          if (igps.getPredicate().getLocalName().equals("linkRela
    tionshipAndRole")) {
265.           Individual role_Individual = ontModel
266.             .getIndividual(igps.getResource().getURI());
267.           String role_Individual_Id = role_Individual.getLocalNa
    me();
268.           String role_Individual_LocalLabel = role_Individual.ge
    tPropertyValue(localLabel)
269.             .toString();
270.           String role_Individual_Shape = role_Individual.getProp
    ertyValue(shape)
271.             .toString();
272.
273.           // 3.1.2 获取关系实例两端的角色实例的属性实例
274.           for (Iterator igp2 = role_Individual.listProperties();
     igp2.hasNext();) {
275.            Statement igps2 = (Statement) igp2.next();
```

```
276.           if (igps2.getPredicate().getLocalName().equals("hasPr
   operty")) {
277.             Individual role_Property_Individual = ontModel
278.               .getIndividual(igps2.getResource().getURI());
279.             for (Iterator igp3 = role_Property_Individual.listPr
   operties(); igp3
280.               .hasNext();) {
281.              Statement igps3 = (Statement) igp3.next();
282.              if (igps3.getPredicate().getLocalName().equals("val
   ue")) {
283.                String role_Property_Individual_Value = igps3.getS
   tring();
284.              }
285.             }
286.            }
287.           }
288.          }
289.         }
290.         ;
291.         // 关系实例的注释属性
292.         String role_Property_Individual_Id = graph_Rela_Individu
   al.getLocalName();
293.         String role_Property_Individual_LocalLabel = graph_Rela_
   Individual
294.           .getPropertyValue(localLabel).toString();
295.         if (graph_Rela_Individual.hasProperty(annotationProperty
   PolyLineLocation)) {
296.          String role_Property_Individual_PolyLineLocation = grap
   h_Rela_Individual
297.            .getPropertyValue(annotationPropertyPolyLineLocation)
   .toString();
298.         }
299.         String role_Property_Individual_LableDisplay = graph_Rel
   a_Individual
300.           .getPropertyValue(lableDisplay).toString();
301.         String role_Property_Individual_Shape = graph_Rela_Indiv
   idual.getPropertyValue(shape)
302.           .toString();
303.         String[] starts = graph_Rela_Individual.getPropertyValue
   (startLocation).toString()
304.           .split(",");
305.         String[] ends = graph_Rela_Individual.getPropertyValue(e
   ndLocation).toString()
306.           .split(",");
```

```
307.
308.         if (graph_Rela_Individual.hasProperty(annotationProperty
    CifString)) {
309.          String role_Property_Individual_CifString = graph_Rela_
    Individual
310.            .getPropertyValue(annotationPropertyCifString).toStri
    ng()
311.            .replace("\\\"", "\"");
312.         }
313.         if (graph_Rela_Individual.getPropertyValue(annotationPro
    pertyAvoidObstructions)
314.           .toString().equals("true")) {
315.
316.         }
317.         if (graph_Rela_Individual.getPropertyValue(annotationPro
    pertyClosestDistance).toString()
318.           .equals("true")) {
319.         }
320.         if (graph_Rela_Individual.hasProperty(annotationProperty
    JumpStatus)) {
321.          String role_Property_Individual_JumpStatus = graph_Rela
    _Individual
322.            .getPropertyValue(annotationPropertyJumpStatus).toStr
    ing();
323.         }
324.         if (graph_Rela_Individual.hasProperty(annotationProperty
    JumpType)) {
325.          String role_Property_Individual_JumpType = graph_Rela_I
    ndividual
326.            .getPropertyValue(annotationPropertyJumpType).toStrin
    g();
327.         }
328.         if (graph_Rela_Individual.getPropertyValue(annotationPro
    pertyJumpReverse).toString()
329.           .equals("true")) {
330.
331.         }
332.         if (graph_Rela_Individual.hasProperty(annotationProperty
    RoutingType)) {
333.          String role_Property_Individual_RoutingType = graph_Rel
    a_Individual
334.            .getPropertyValue(annotationPropertyRoutingType).toSt
    ring();
335.         }
```

```
336.         if (graph_Rela_Individual.hasProperty(annotationProperty
   Smooth)) {
337.          String role_Property_Individual_Smooth = graph_Rela_Ind
   ividual
338.            .getPropertyValue(annotationPropertySmooth).toString(
   );
339.         }
340.         if (graph_Rela_Individual.hasProperty(annotationProperty
   Text)) {
341.          String[] texts = graph_Rela_Individual.getPropertyValue
   (annotationPropertyText)
342.            .toString().split(",");
343.         }
344.         if (graph_Rela_Individual.hasProperty(annotationProperty
   SdIdentification)) {
345.          String role_Property_Individual_SdIdentification = grap
   h_Rela_Individual
346.            .getPropertyValue(annotationPropertySdIdentification)
   .toString();
347.         }
348.         if (graph_Rela_Individual.hasProperty(annotationProperty
   Color)) {
349.          String role_Property_Individual_Color = graph_Rela_Indi
   vidual
350.            .getPropertyValue(annotationPropertyColor).toString()
   ;
351.         }
352.         Integer role_Property_Individual_ModelSize = Integer.par
   seInt(
353.           graph_Rela_Individual.getPropertyValue(annotationPrope
   rtyModelSize).toString());
354.
355.         // 3.2 获取关系实例的属性实例及相关信息
356.         for (Iterator igp = graph_Rela_Individual.listProperties
   (); igp.hasNext();) {
357.          Statement igps = (Statement) igp.next();
358.          if (igps.getPredicate().getLocalName().equals("hasPrope
   rty")) {
359.           // 获取关系实例的属性实例
360.           Individual rela_Property_Individual = ontModel
361.             .getIndividual(igps.getResource().getURI());
362.           // 找到属性值
363.           for (Iterator igp2 = rela_Property_Individual.listProp
   erties(); igp2
```

```
364.             .hasNext();) {
365.           Statement igps2 = (Statement) igp2.next();
366.           if (igps2.getPredicate().getLocalName().equals("value
   ")) {
367.            String rela_Property_Individual_value = igps2.getStr
   ing();
368.           }
369.          }
370.         }
371.        }
372.       }
373.       // 4. 获取模型的约束实例及对应的信息(GOPPRR-E 的 E)
374.       if (igs.getPredicate().getLocalName().equals("graphIncludingConnector")) {
375.        // 获取模型的约束实例
376.        Individual connetor_Individual = ontModel.getIndividual(igs.getResource().getURI());
377.        String connectPoint = null;
378.        String connectObject = null;
379.        String connectRelationship = null;
380.        String connectRole = null;
381.        for (Iterator igp = connetor_Individual.listProperties(); igp.hasNext();) {
382.         Statement igps = (Statement) igp.next();
383.         if (igps.getPredicate().getLocalName().equals("linkFromRelationship")) {
384.          connectRelationship = igps.getResource().getLocalName();
385.          System.out.println(connectRelationship);
386.         } else if (igps.getPredicate().getLocalName().equals("linkRelationshipAndRole")) {
387.          connectRole = igps.getResource().getLocalName();
388.          System.out.println(connectRole);
389.         } else if (igps.getPredicate().getLocalName().equals("linkToObject")) {
390.          connectObject = igps.getResource().getLocalName();
391.          System.out.println(connectObject);
392.         } else if (igps.getPredicate().getLocalName().equals("linkObjectAndPoint")) {
393.          connectPoint = igps.getResource().getLocalName();
394.          System.out.println(connectPoint);
395.         }
396.        }
397.        if (connectPoint == null) {
```

```
398.          // 约束方式
399.//           connectRelationship + "." + connectRole, connectObje
   ct;
400.          System.out.println(connectObject + "," + connectRelatio
   nship + "." + connectRole);
401.         } else {
402.          // 约束方式
403.//           connectRelationship + "." + connectRole,connectObjec
   t + "." + connectPoint
404.          System.out.println(connectRelationship + "." + connectR
   ole + "," + connectObject
405.            + "." + connectPoint);
406.         }
407.        }
408.       }
409.      }
410.     }
411.    }
412.   }
413. }
```

10.3 案例

通过调用 getModel 方法获取的信息如下所示：

其中 WgVEshCQ 为 Block_Defintion_Diagram 图元模型下的实例，是模型的 id 属性，也就是工具中 model 目录下面的具体模型，A1 为该模型的 locallabel 属性；

由工具可知：

（1） 该模型有两个对象实例(graphIncludingObject)，也就是MetaGraph工具中的两个模块，其中“模块”为对象实例的locallabel属性

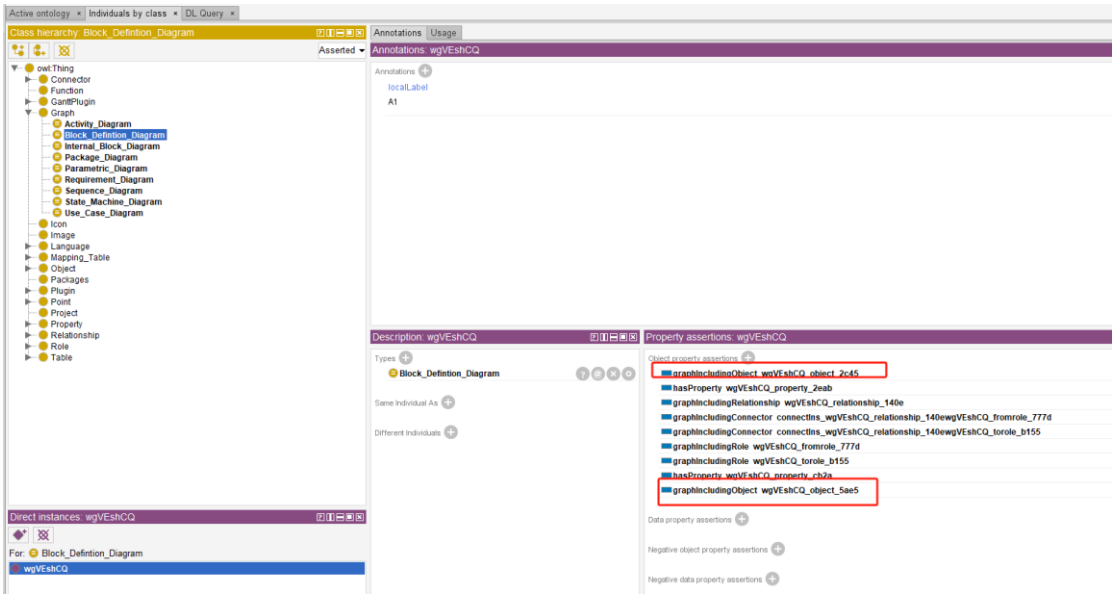

图 10-2对象实例（本体）

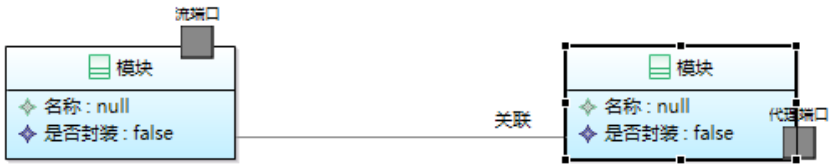

图 10-3模型中的对象实例（MetaGraph）

（2） 对象实例有两个属性实例，也就是模块中的名称以及是否封装，“名

称”，“是否封装”为属性实例的localabel属性，后面的值为具体的属性值

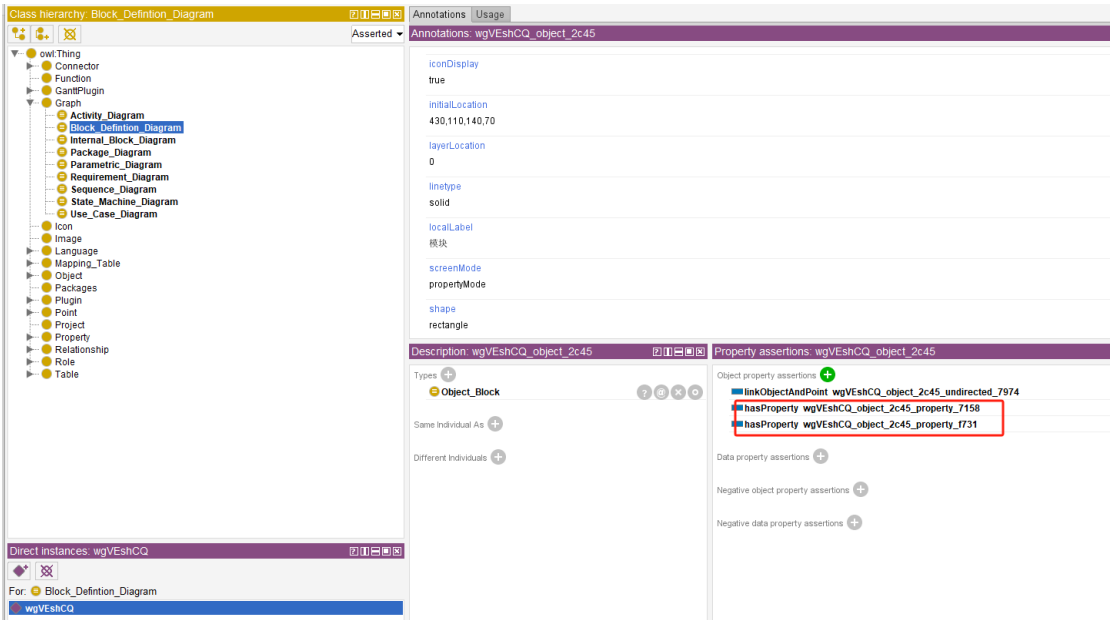

图 10-3属性实例（本体）

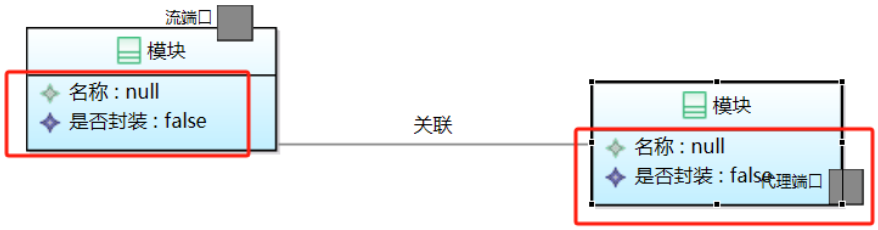

图 10-4对象实例的属性实例（MetaGraph）

（3） 对象实例有一个点实例，也就是模块中的流端口，“流端口”为点实例的locallabel属性，双击流端口可以看到点实例的属性实例以及属性值

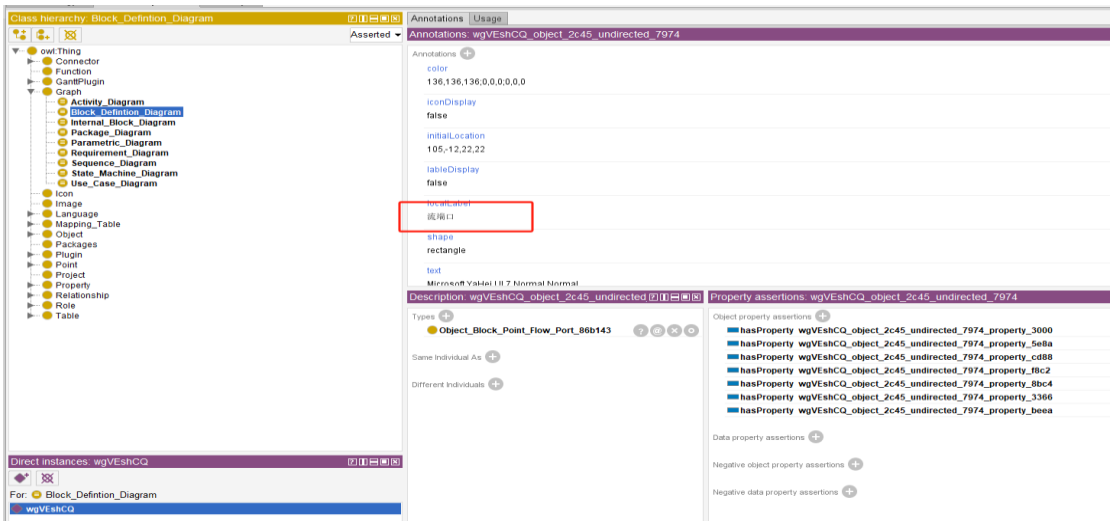

图 10-5点实例（本体）

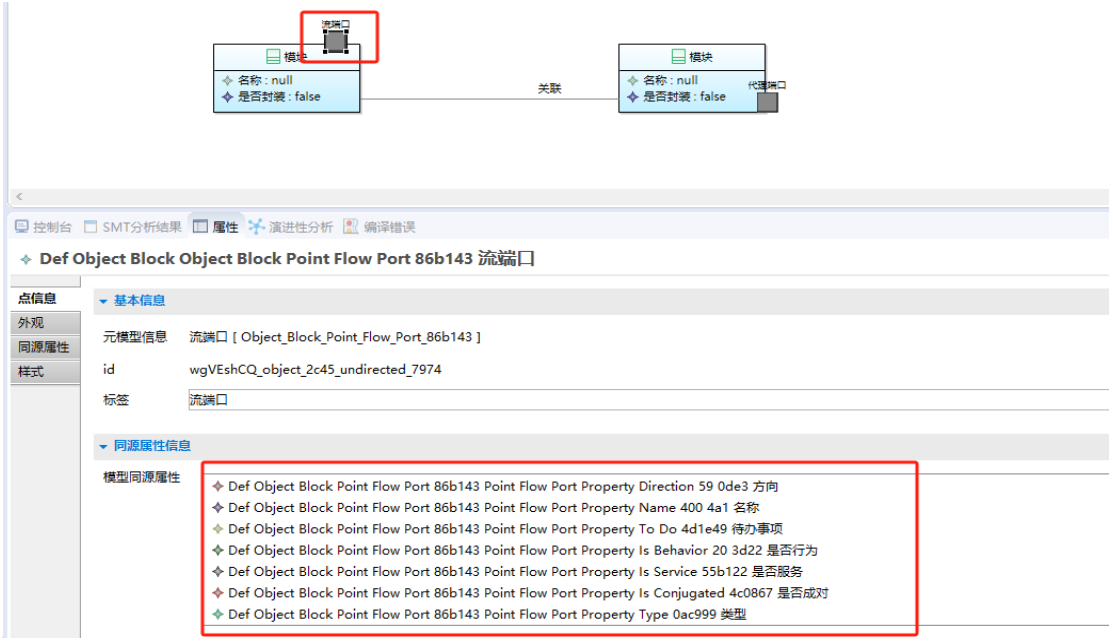

图 10-6点实例的属性实例

（4） 关联为模型的关系实例，由工具可知关系实例的属性实例localLabel为名称；以及关系实例的角色实例角色A和角色B;上下文为关系实例两端的角色实例的属性实例

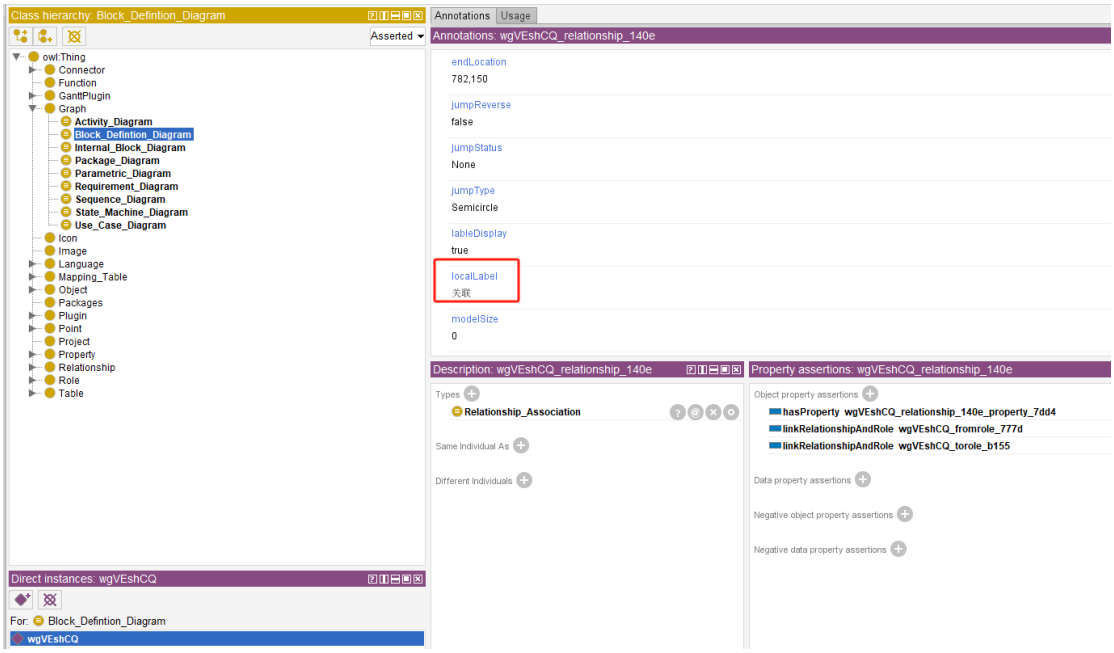

图 10-7对象实例的关系实例

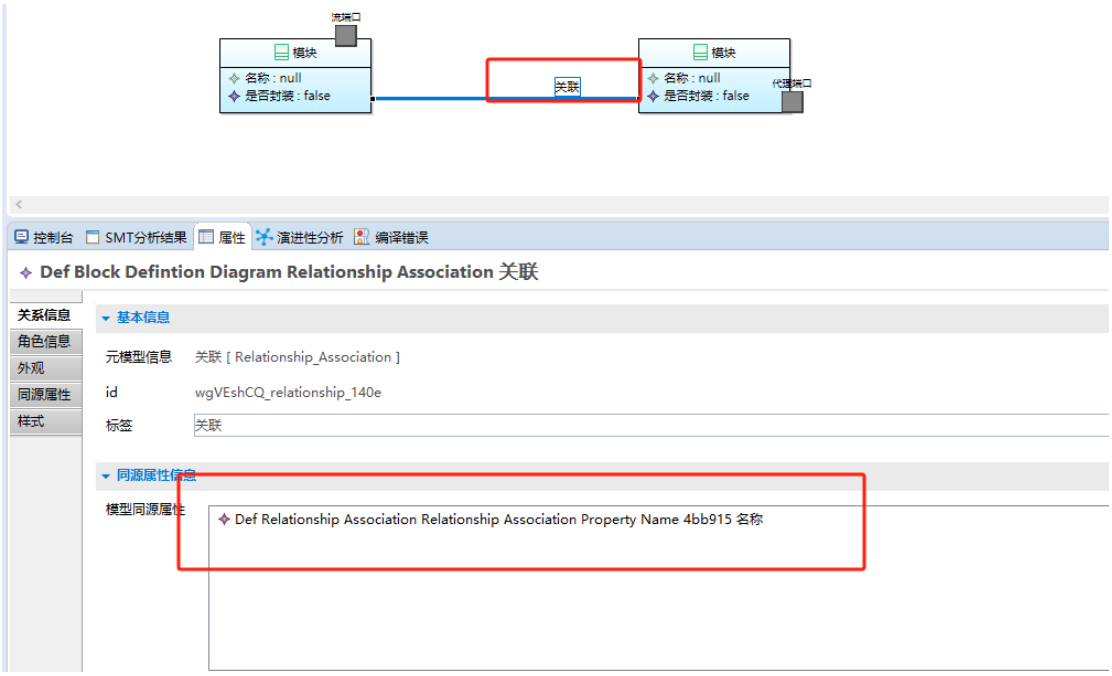

图 10-8关系实例的属性实例

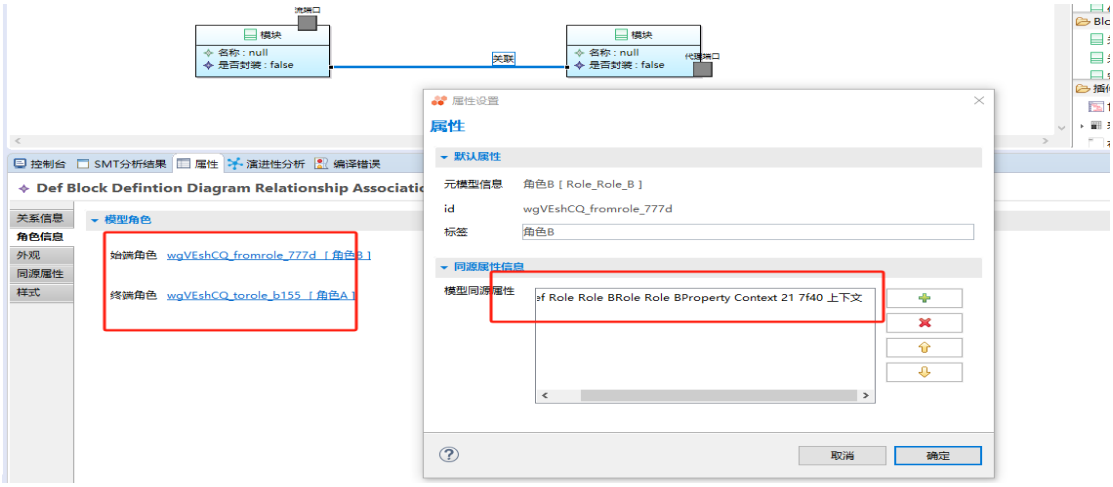

图 10-9关系实例的角色实例以及角色实例的属性实例

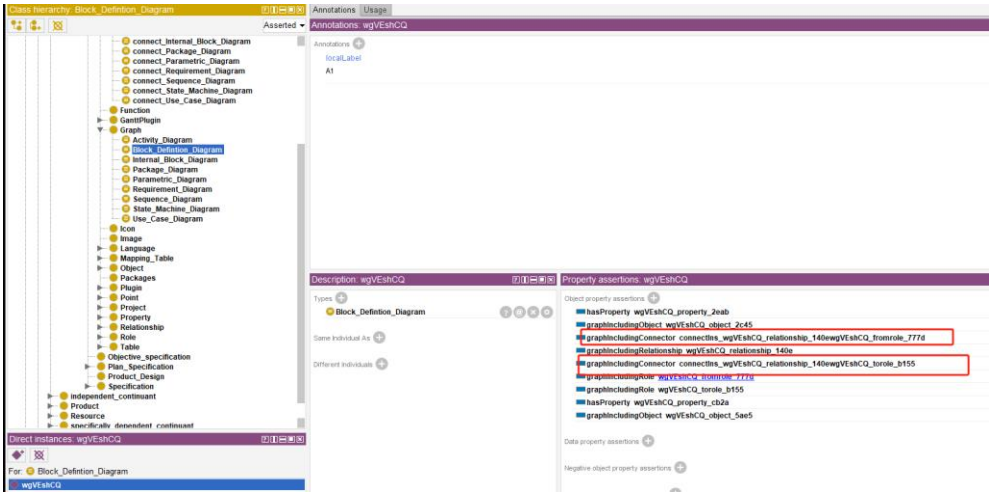

图 10-10模型实例的约束实例及对应的信息