第二讲 知识表示与知识建模

大纲

- 早期知识表示简介
- 基于语义网的知识表示框架
  - RDF和RDFS
  - OWL和OWL2 Fragments
  - SPARQL查询语言
  - Json-LD、RDFa、HTML5 MicroData等新型知识表示
- 典型知识库项目的知识表示
- 基于本体工具 (Protégé)的知识建模实践

第一部分:早期知识表示简介

知识表示的重要性

- 知识是智能的基础
  - 机器可以获得知识
  - 机器可以运用知识

- 符合计算机要求的知识模式
  - 计算机能存储、处理的知识表示模式
  - 数据结构 (List, Table, Tree, Graph, etc.)

知识的特性

- 相对正确性
  - 一定条件下
  - 某种环境中
  - …
- 不确定性
  - 存在“中间状态”
  - “真”(“假”)程度
  - 随机性
  - 模糊性
  - 经验性
  - 不完全性
  - …
- 可表示性&可利用性
  - 语言
  - 文字
  - 图形
  - 图像
  - 视频
  - 音频
  - 神经网络
  - 概率图模型
  - …

知识分类

- 常识性知识、领域性知识（作用范围）
- 事实性知识、过程性知识、控制知识（作用及表示）
- 确定性知识、不确定性知识（确定性）
- 逻辑性知识、形象性知识（结构及表现形式）

早期知识表示方法

- 一阶谓词逻辑 (First-Order Logic)
- 产生式规则 (Production Rule)
- 框架 (Framework)
- 语义网络 (Semantic Network)
- 逻辑程序 (Logic Programming)
- 缺省逻辑 (Default Logic)
- 模态逻辑 (Modal Logic)

早期知识表示简介

一阶谓词逻辑

优点

- 自然性
- 接近自然语言，容易接受
- 精确性
- 用于表示精确知识
- 严密性
- 有严格的形式定义和推理规则
- 易实现性
- 易于转换为计算机内部形式

缺点

- 无法表示不确定性知识
- 难以表示启发性知识及元知识
- 组合爆炸，经常出现事实、规则等的组合爆炸
- 效率低，推理复杂度通常较高

早期知识表示简介

Horn逻辑：一阶谓词逻辑的子集

- 表达形式简单，复杂度低；著名的Prolog语言就是基于Horn逻辑设计实现的。

早期知识表示简介

Horn逻辑：一阶谓词逻辑的子集

- 原子Atom

$$p(t_1, t_2 ..., t_n)$$

  - p 是谓词，n是目，t_i 是项（变量或者常量）
  - 例子: has_child(Helen, Jack)

早期知识表示简介

Horn逻辑：一阶谓词逻辑的子集

- 规则*Rules* 由原子构建:

$$H :\!- B_1, B_2, ..., B_m.$$

  - H与$B_1, B_2, ..., B_m$是原子
  - H是头部原子
  - $B_1, B_2, ..., B_m$是体部原子

has_child(X, Y) :- has_son(X, Y)

- 事实是没有体部且没有变量的规则:

$$F(v_1, v_2, ..., v_n) :-$$

has_child(Helen, Jack) :-

早期知识表示简介

描述逻辑：一阶谓词逻辑的可判定子集

- 用于描述概念，属性；对于术语知识库的构建提供了便捷的表达形式。

早期知识表示简介

描述逻辑系统

概念和关系

- 概念——解释为一个领域的子集

  示例：学生，已婚者：

  {x| Student(x) }，{x| Married(x) }

- 关系——解释为指该领域上的二元关系(笛卡尔乘积)

  示例：朋友，爱人：

  {<x,y> | friend(x,y) }，{<x,y> | loves(x,y) }

- 个体——一个领域内的实例

  示例：小明，小红：

  {Ming, Hong}

早期知识表示简介

描述逻辑的知识库O:=<T， A>，T即Tbox，A即Abox

- Tbox

Tbox包含内涵知识，描述概念的一般性质。由于概念之间存在包含关系，Tbox知识形成类似格的结构，这种数学结构是由包含关系决定的，与具体实现无关；

TBox语言

- 定义: 引入概念以及关系的名称

  Mother, Person, has_child

- 包含：声明包含关系的公理

  Mother ⊑ ∃ has_child.Person

概念和概念，关系和概念之间可以组成更复杂的概念。

早期知识表示简介

描述逻辑的知识库O:=<T， A>，T即Tbox，A即Abox

- Abox

Abox包含外延知识 (又称断言知识)，描述论域中的特定个体。

ABox语言

- 概念断言——表示一个对象是否属于某个概念

  Mother(Helen), Person(Jack)

- 关系断言——表示两个对象是否满足一定的关系

  has_child(Helen, Jack)

早期知识表示简介

产生式系统

- 是一种更广泛意义的规则系统，和谓词逻辑有关联，也有区别；

- 早期的专家系统多数是基于产生式系统：

  Feigenbaum研制的化学分子结构专家系统DENDRAL

  Shortliffe研制的的诊断感染性疾病的专家系统MYCIN

  …

早期知识表示简介

产生式模型

$P \rightarrow Q$ 或

IF P THEN Q *CF* = *[0, 1]*

- 其中，P是产生式的前提，Q是一组结论或操作，CF (Certainty Factor)为确定性因子，也称置信度。
- 谓词逻辑中的规则与产生式的基本形式相似，事实上，蕴涵式只是产生式的一种特殊情况。理由如下：
  - 谓词逻辑规则只能表示精确知识，其值非“真”即“假”，而产生式不仅可以表示精确知识，而且还可以表示不精确知识；
  - 用产生式表示知识的系统中，“事实”与产生式的“前提”中所规定的条件进行匹配时，可以是“精确匹配”，也可以是基于相似度的“不精确匹配”，只要相似度落入某个预先设定的范围内，即可认为匹配。但对谓词逻辑的规则而言，其匹配必须是精确的。

早期知识表示简介

产生式例子

IF　本微生物的染色斑是革兰氏阴性

　　本微生物的形状呈杆状

　　病人是中间宿主

THEN　该微生物是绿脓杆菌，置信度为CF=0.6

CF表示知识的强度，谓词逻辑中的规则不可以这样做。

早期知识表示简介

产生式系统优点

- 自然性：由于产生式系统采用了人类常用的表达**因果关系**的知识表示形式，既直观、自然，又便于进行推理。
- 模块性：产生式系统中的规则形式相同，易于模块化管理。
- 有效性：能表示确定性知识、不确定性知识、启发性知识、过程性知识等。
- 清晰性：产生式有固定的格式，既便于规则设计，又易于对规则库中的知识进行一致性、完整性检测。

早期知识表示简介

产生式系统缺点

- 效率不高

  产生式系统求解问题的过程是一个反复进行“匹配—冲突消解—执行”的过程。由于规则库一般都比较庞大，而匹配又是一件十分费时的工作，因此，其工作效率不高。此外，在求解复杂问题时容易引起组合爆炸。

- 不能表达具有结构性的知识

  产生式系统对具有结构关系的知识无能为力，它不能把具有结构关系的事物间的区别与联系表示出来，因此，人们经常将它与其它知识表示方法（如框架表示法、语义网络表示法）相结合。

早期知识表示简介

框架 (Framework)

- 框架理论的基本思想：认为人们对现实世界中各种事物的认识都是以一种类似于框架的结构存储在记忆中。

- 当面临一个新事物时，就从记忆中找出一个合适的框架，并根据实际情况对其细节加以修改、补充，从而形成对当前事物的认识。

早期知识表示简介

框架基本组成

- 框架：是一种描述对象(事物、事件或概念等)属性的数据结构，在框架理论中，框架是知识表示的基本单位。
- 一个框架由若干个“槽” (Slot)结构组成，每个槽又可分为若干个“侧面”。

  一个槽：用于描述所论对象某一方面的属性；

  一个侧面：用于描述相应属性的一个方面。
- 槽和侧面所具有的属性值分别称为槽值和侧面值。

早期知识表示简介

框架基本组成

<框架名>

槽名1：侧面名1　　值1，值2，...，值p1

　　侧面名2　　值1，值2，...，值p2

　　侧面名m1　　值1，值2，...，值pm1

槽名n：侧面名1　　值1，值2，...，值r1

约束：　约束条件1

　　约束条件n

早期知识表示简介

一个框架示例

框架名：	t intent result案
犯罪意图：	intent
犯罪结果：	result
被杀者：	y
知情人：	$\{ z_i \mid i \in I\}$
罪犯：	t
条件一：	有某个 z_i 指控 t
条件二：	t 招认

早期知识表示简介

利用上页框架在知识库中匹配得到实例

框架名：	小A抢劫杀人案
犯罪意图：	抢劫
犯罪结果：	杀人
被杀者：	小B
知情人：	小C
罪犯：	小A
条件一：	有小C指控小A
条件二：	小A招认

早期知识表示简介

框架的优缺点

- 优点

  框架对于知识的描述非常完整和全面；基于框架的知识库质量非常高；且框架允许数值计算，这一点优于其它知识表示语言；

- 缺点

  框架的构建成本非常高，对知识库的质量要求非常高；框架的表达形式不灵活，很难同其它形式的数据集相互关联使用。

早期知识表示简介

语义网络 (Semantic Network)

- 1968年J.R.Quillian在其博士论文中最先提出语义网络，把它作为人类联想记忆的一个显式心理学模型，并在他设计的可教式语言理解器TLC (Teachable Language Comprehenden)中用作知识表示方法。

- 语义网络的基本思想：在网络中，用“节点”代替概念，用节点间的“连接弧”(称为联想弧)代替概念之间的关系，因此，语义网络又称**联想网络**。它在形式上是一个带标识的有向图。由于所有的概念节点均通过联想弧彼此相连，Quillian希望他的语义网络能用于知识推导。

早期知识表示简介

语义网络基本形式

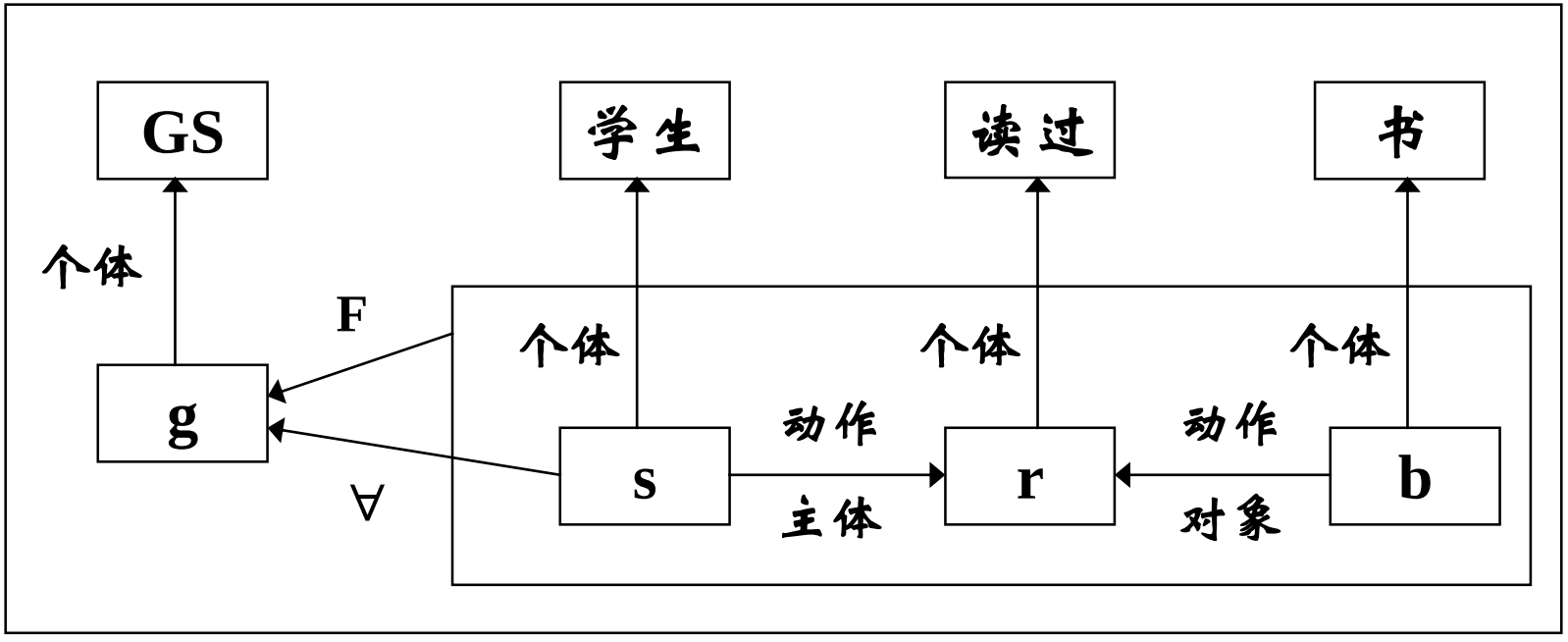
- 语义网络中的节点：表示各种事物、概念、情况、属性、动作、状态等，每个节点可以带有若干属性，一般用框架或元组表示。此外，节点还可以是一个语义子网络，形成一个多层次的嵌套结构。
- 语义网络中的弧：表示各种语义联系，指明它所连接的节点间某种语义关系。
- 节点和弧都必须带有标识，以便区分各种不同对象以及对象间各种不同的语义联系。最简单的语义网络是一个三元组：

（节点1，弧，节点2）

早期知识表示简介

语义网络例子：“每个学生都读过一本书”

- 用谓词逻辑表示：(∀s)学生(s) (∃b) 书(b)[读过(s,b)]
- 用语义网络表示：



本质上将逻辑运算符和逻辑项映射到了图中的元素。

早期知识表示简介

语义网络优点

- **结构性**：语义网络是一种结构化的知识表示方法，它能把事物的属性以及事物间的各种语义联想显式地表示出来。
- **联想性**：它最初是作为人类联想记忆模型提出来的。
- **自然性**：直观地把事物的属性及其语义联系表示出来，便于理解，自然语言与语义网络的转换比较容易实现，故语义网络表示法在自然语言理解系统中的应用最为广泛。

早期知识表示简介

语义网络缺点

- **非严格性：**与一阶谓词逻辑相比，语义网络没有公认的形式表示体系。一个给定的语义网络所表达的含义完全依赖于处理程序如何对它进行解释。通过推理网络而实现的推理不能保证其正确性。此外，目前采用的表示量词 (包括全称量词和存在量词)的语义网络表示法在逻辑上是不充分的，不能保证不存在二义性。

- **处理上的复杂性：**语义网络表示知识的手段多种多样，虽然灵活性很高，但同时也由于表示形式的不一致使得对其处理的复杂性提高，对知识的检索也就相对复杂，要求对网络的搜索要有强有力的组织原则。

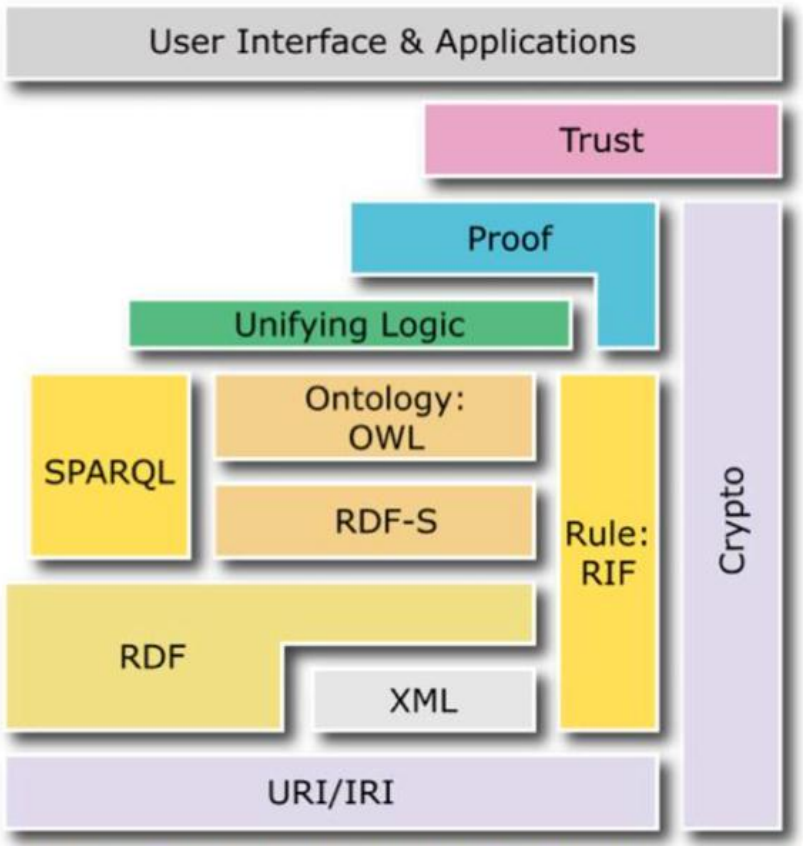
第二部分：基于语义网的知识表示框架

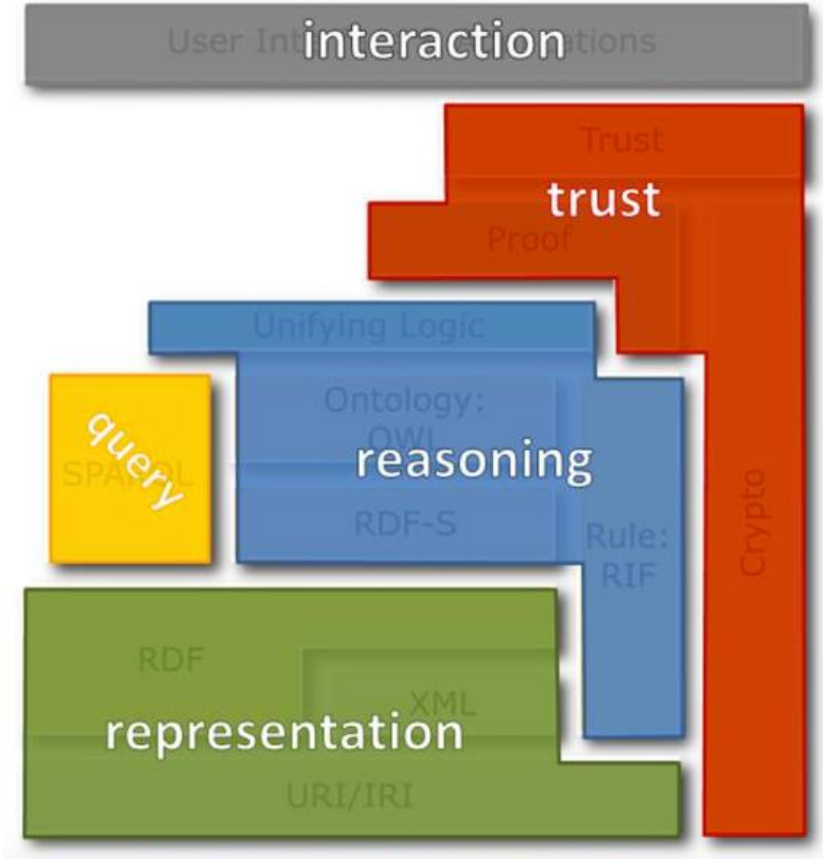
语义网 Semantic Web

我有两个梦想：第一个是连接世界上的每一个人。现在这个梦想已经实现了。

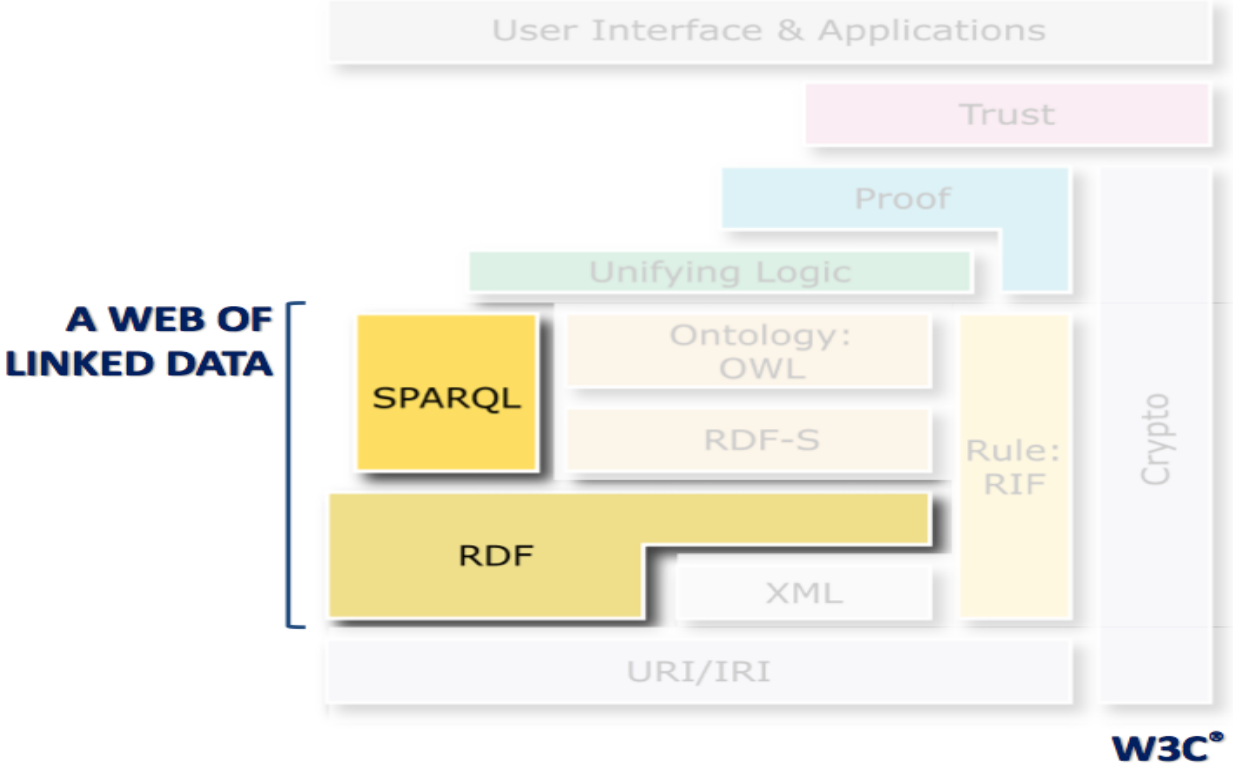
我的第二个梦想是连接世界上的每一个事物。这个光荣的使命交给了语义网！

W3C推荐的语义网标准栈





知识图谱所需使用的主要技术标准



principles

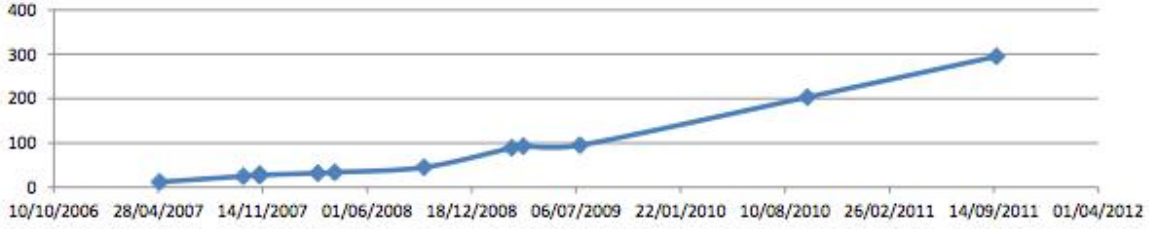
- Use RDF as data format
- Use URIs as names for things
- Use HTTP URIs so that people can look up those names
- When someone looks up a URI, provide useful information (RDF, HTML, etc.) using content negotiation
- Include links to other URIs so that related things can be discovered

May 2007

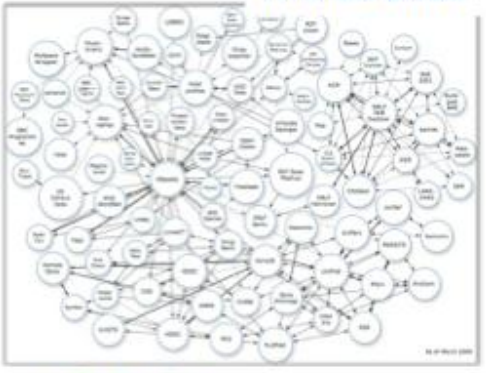
April 2008

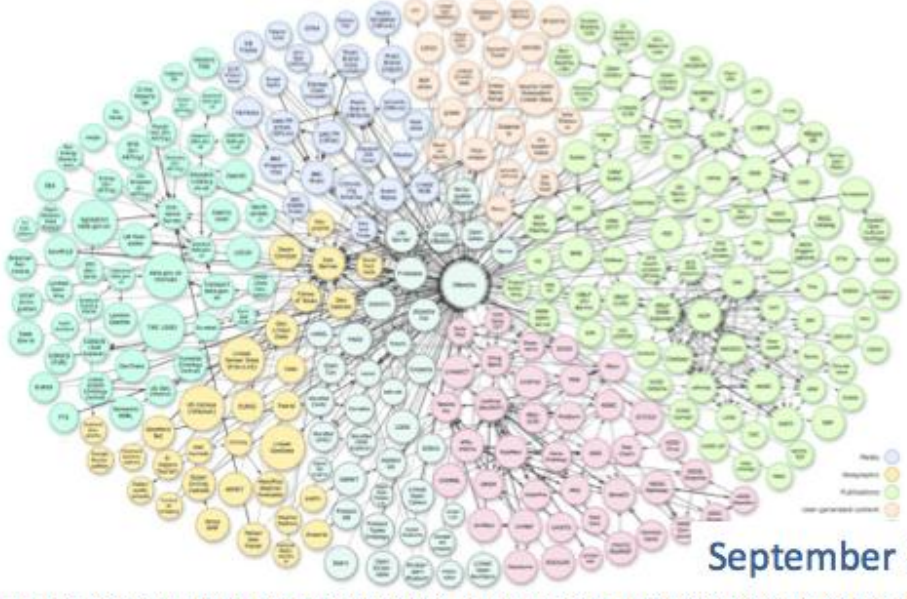
September 2008

Linking Open Data

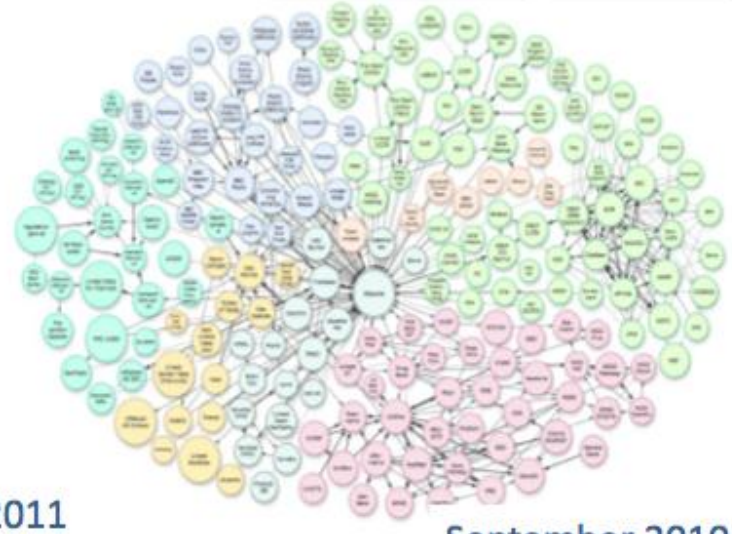


March 2009





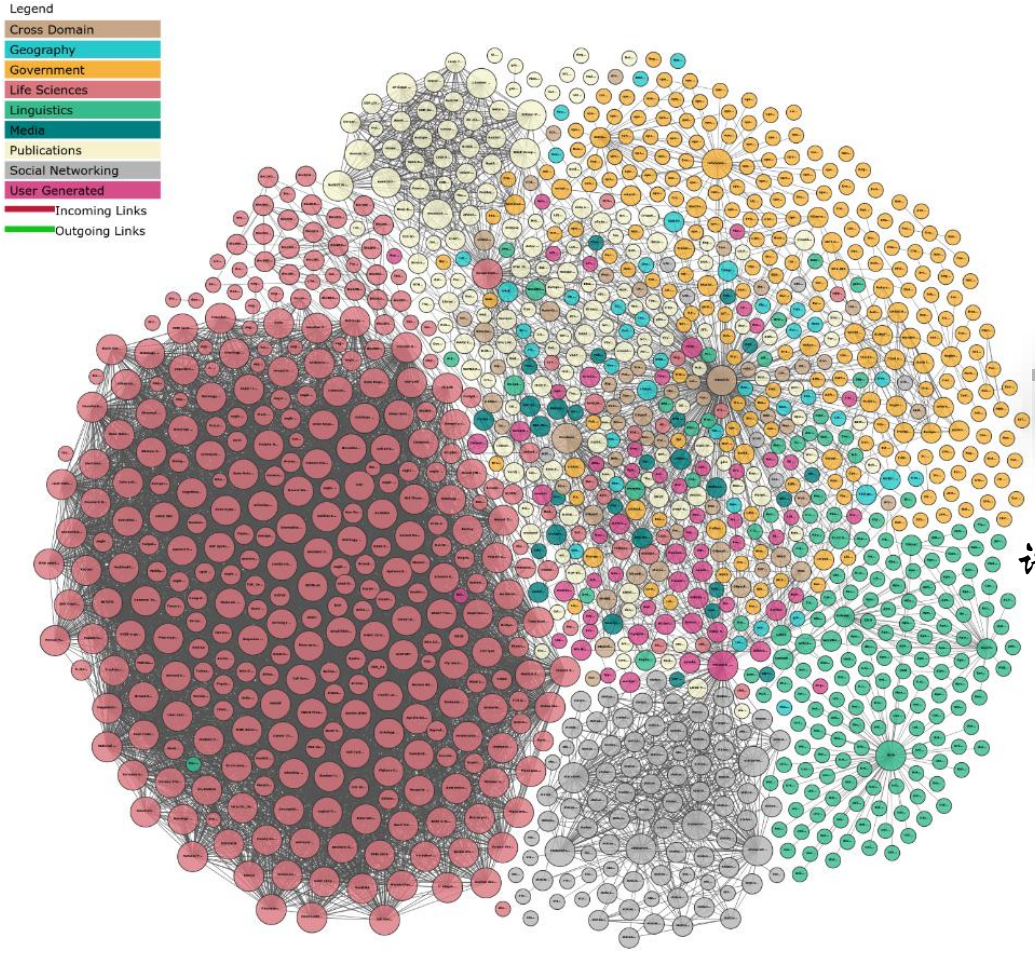
September 2011

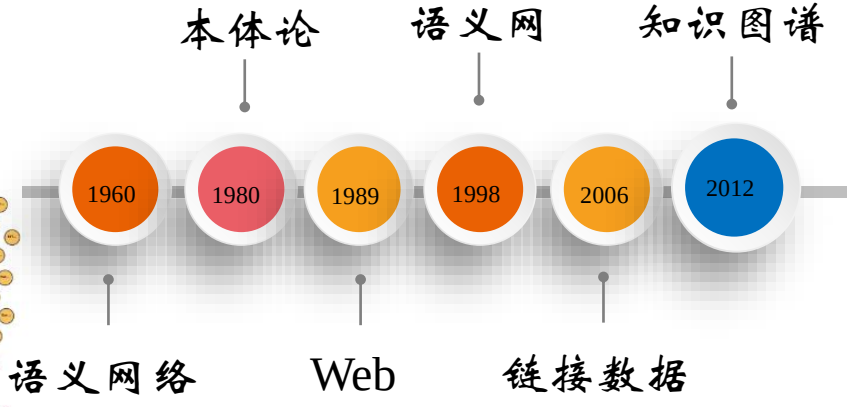


September 2010

Linking Open Data cloud diagram, by Richard Cyganiak and Anja Jentzsch. http://lod-cloud.net/

知识图谱 (Knowledge Graph, KG)



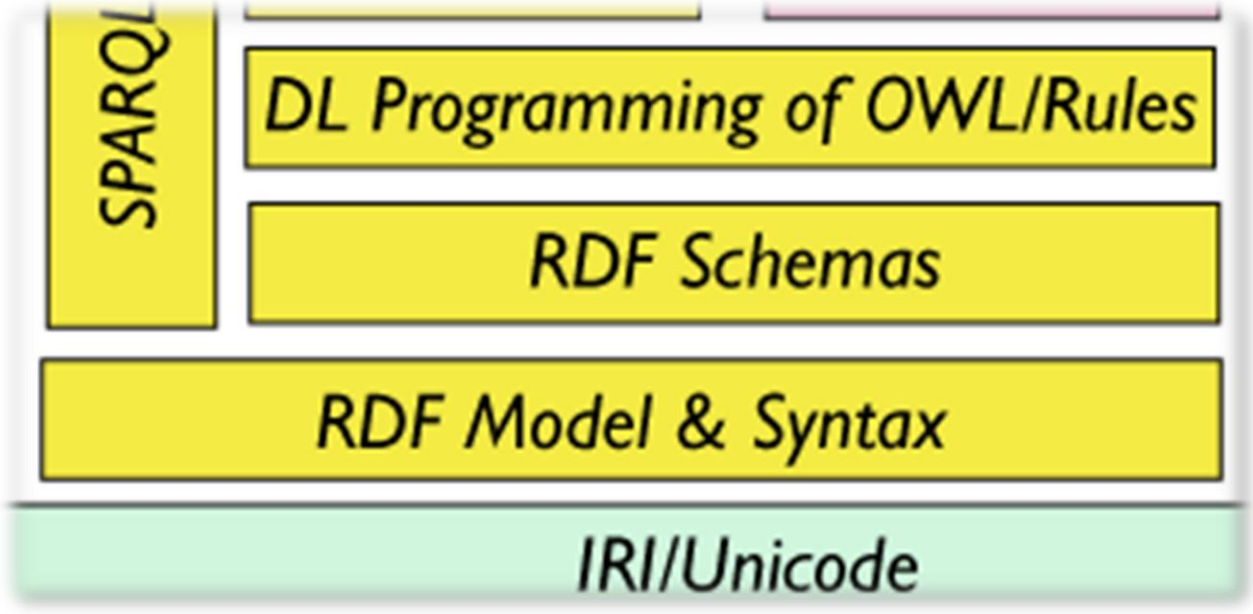


"Linking Open Data cloud diagram 2017, by Andrejs Abele, John P. McCrae, Paul Buitelaar, Anja Jentzsch and Richard Cyganiak. http://lod-cloud.net/"

RDF简介

RDF代表

Resource Description Framework（资源描述框架）



RDF简介

RDF 代表

Resource: 页面、图片、视频等任何具有URI标识符

Description: 属性、特征和资源之间的关系

Framework: 模型、语言和这些描述的语法

RDF模型

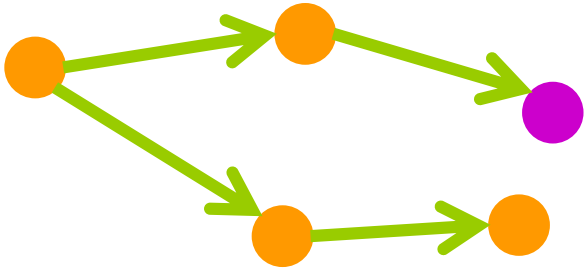
- 在RDF中，知识总是以三元组形式出现
- RDF是一个三元组 (triple) 模型，即每一份知识可以被分解为如下形式：
- (subject (主) , predicate (谓) , object (宾))

RDF示例

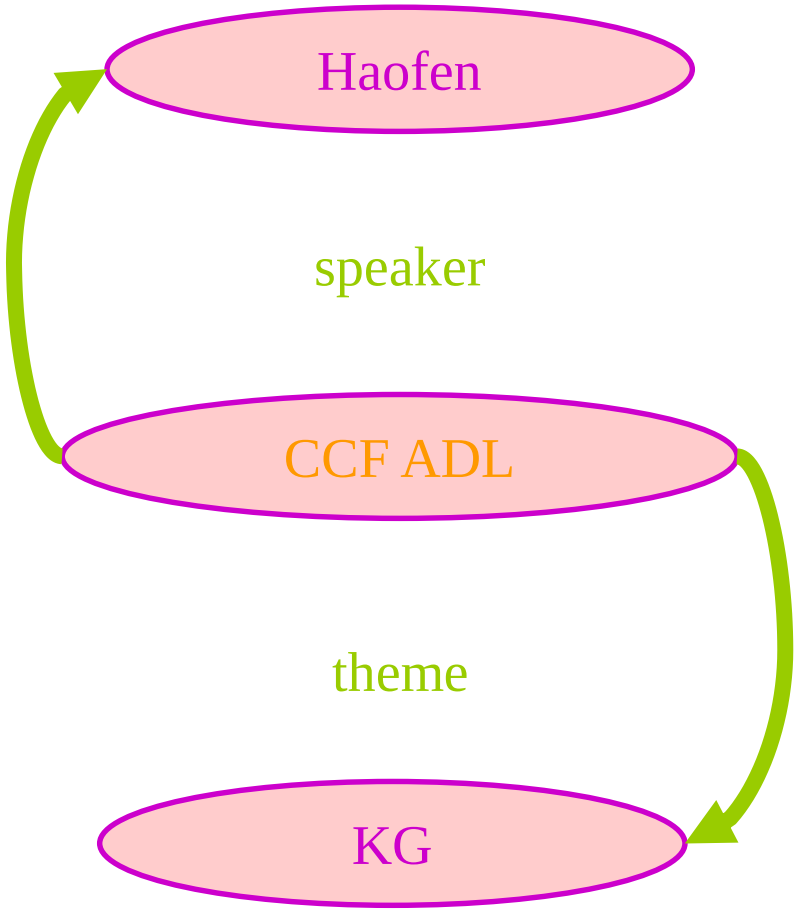
- CCF ADL邀请王昊奋作为讲者，演讲主题是知识图谱

- (CCF ADL, speaker, Haofen)
  (CCF ADL, theme, KG)

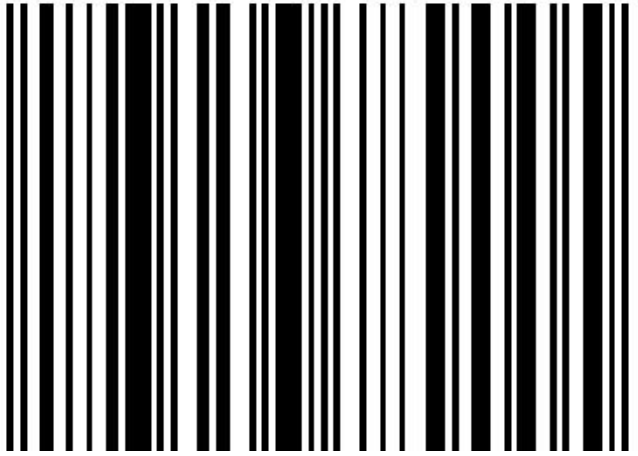
RDF is also a graph model
to link the descriptions of resources



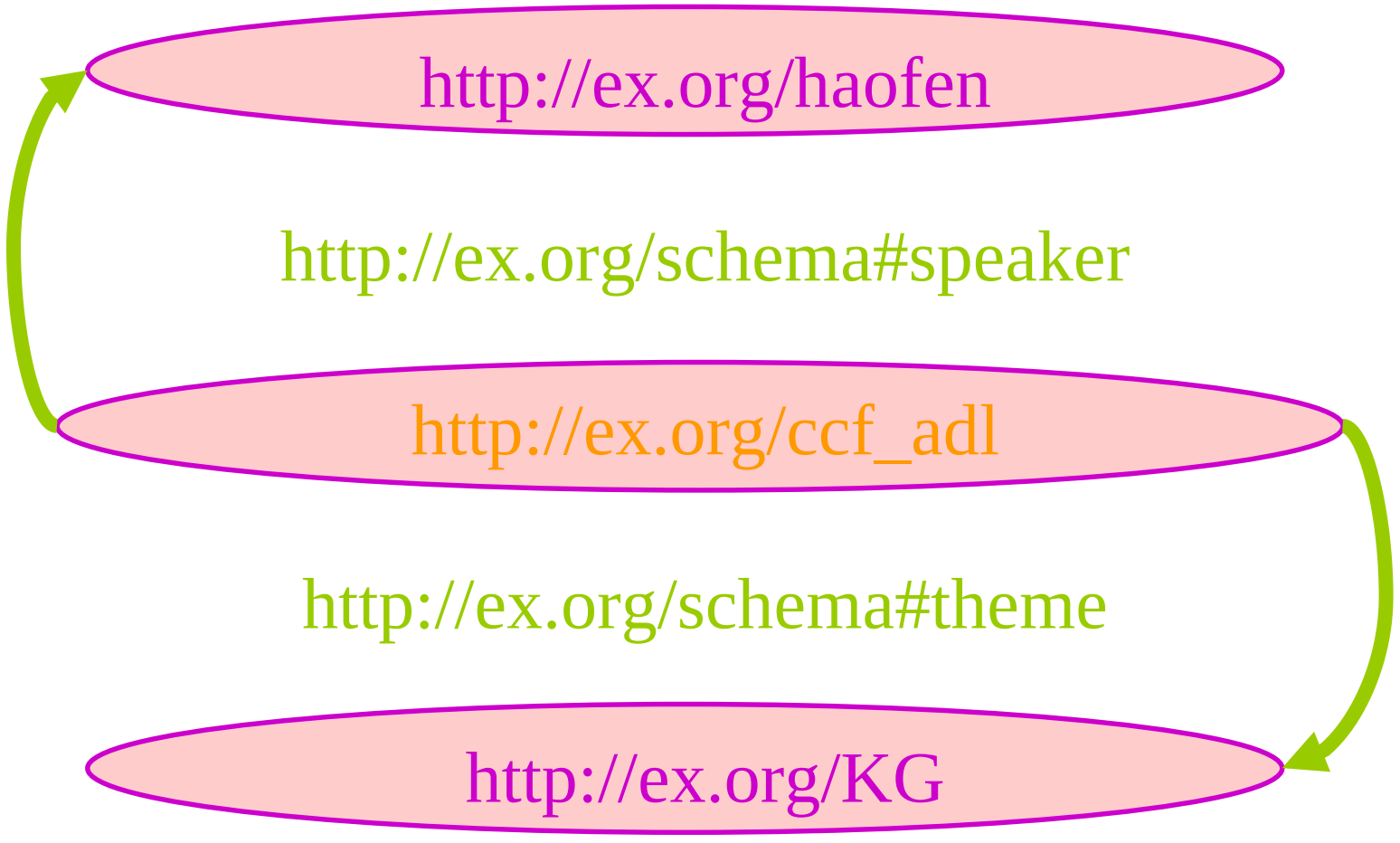
RDF triples can be seen as arcs of a graph (vertex, edge, vertex)

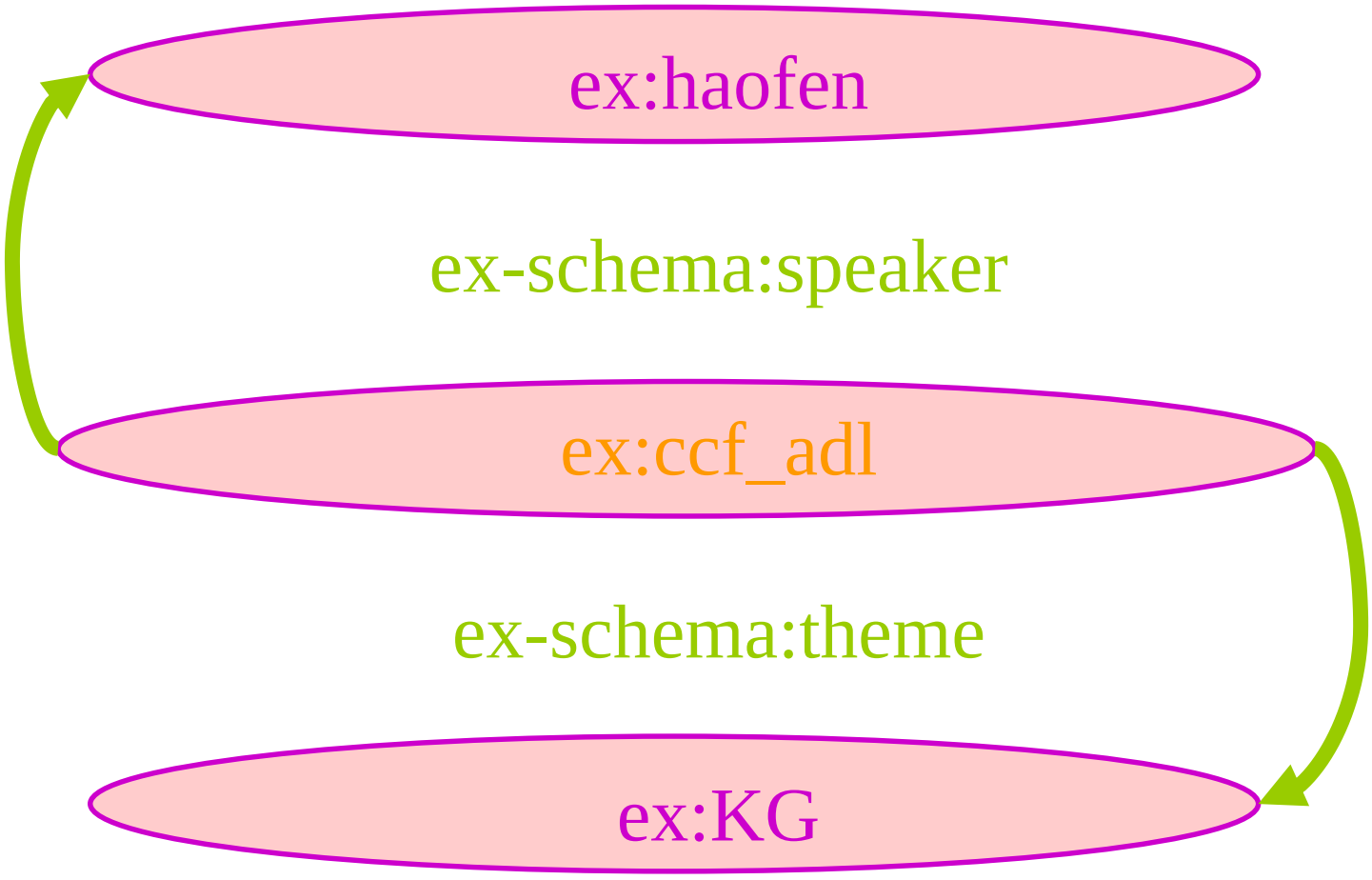

Haofen
speaker
CCF ADL
theme
KG

in RDF resources and properties are identified by URIs



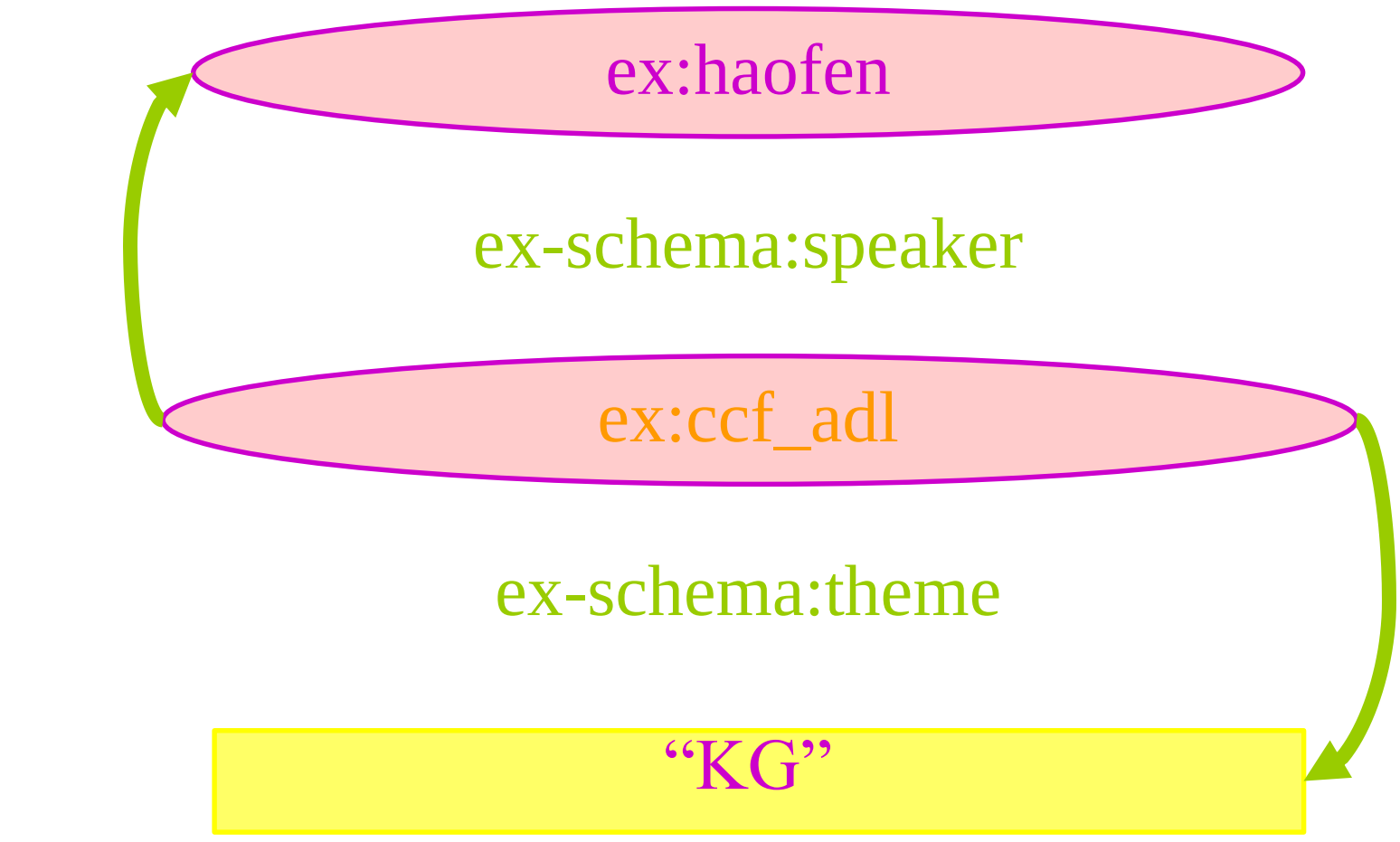
http://mydomain.org/mypath/myresource


http://ex.org/haofen
http://ex.org/schema#speaker
http://ex.org/ccf_adl
http://ex.org/schema#theme
http://ex.org/KG


ex:haofen
ex-schema:speaker
ex:ccf_adl
ex-schema:theme
ex:KG

in RDF values of properties can also be literals i.e. strings of characters

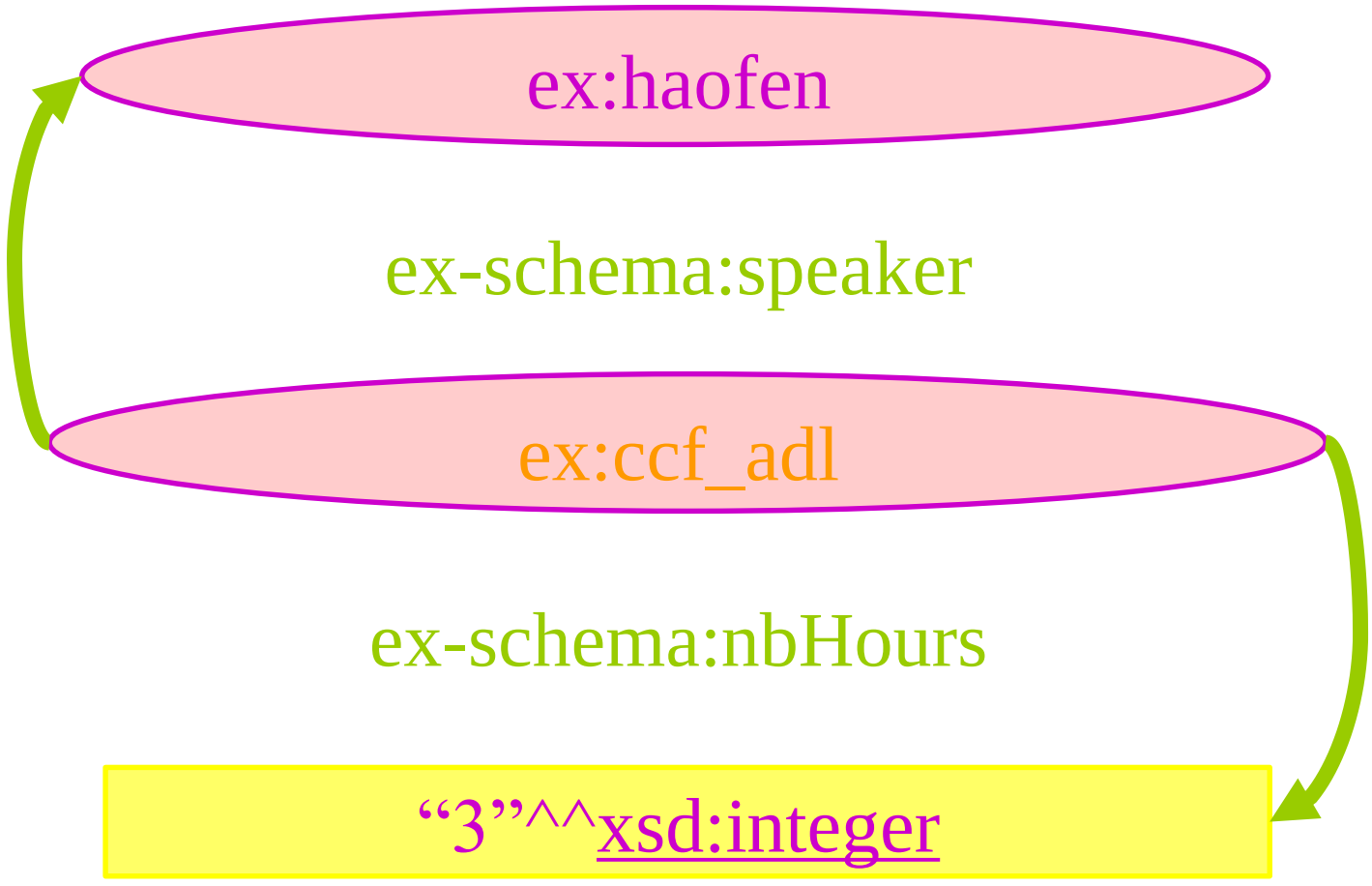
(CCF ADL, speaker, Haofen)
(CCF ADL, theme, “KG”)


ex:haofen
ex-schema:speaker
ex:ccf_adl
ex-schema:theme
“KG”

in RDF literal values of properties can also be typed with XML datatypes

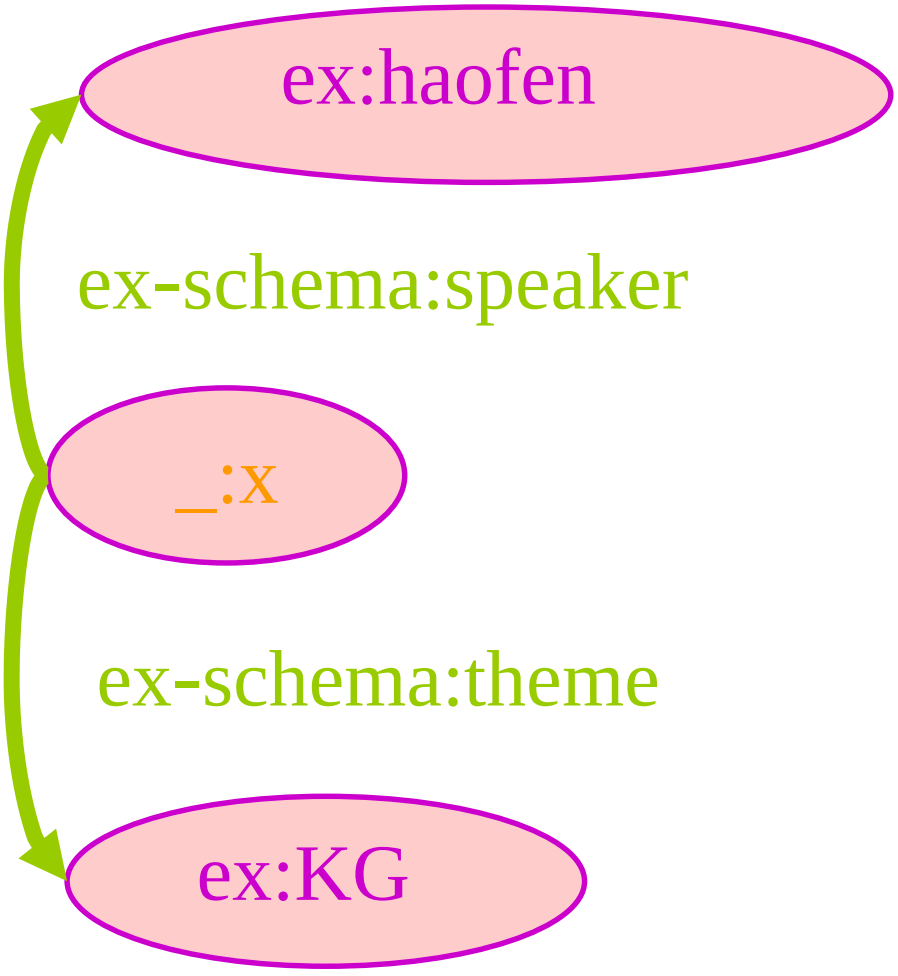
CCF ADL has one speaker Haofen

and last 3 hours


ex:haofen
ex-schema:speaker
ex:ccf_adl
ex-schema:nbHours
“3”^^xsd:integer

RDF空白节点 (Blank Nodes)

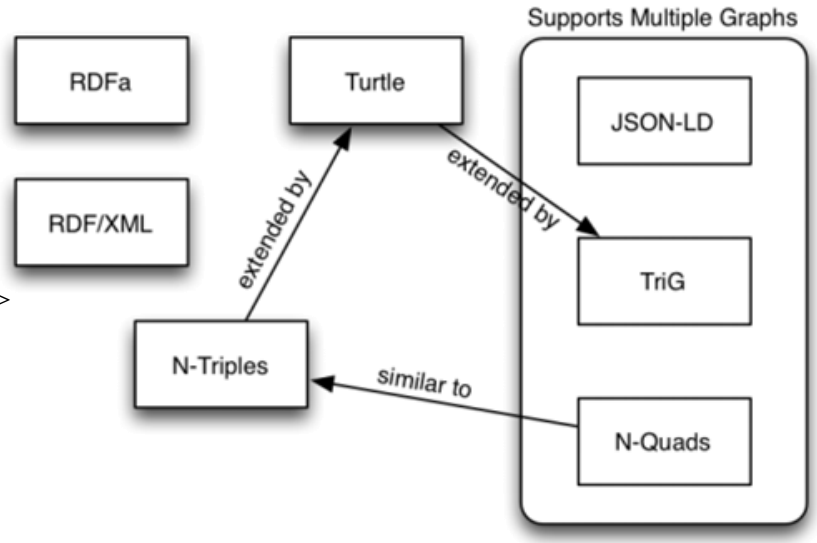
- RDF允许空白节点
- 一个资源可以是匿名的
- 即不被URI标识，并标识为_: xyz
- 例如：Haofen是某一次KG讲座的讲者


ex:haofen
ex-schema:speaker
_:x
ex-schema:theme
ex:KG

RDF是数据模型，不是序列化格式

- **RDF/XML**

```
<rdf:RDF
  xmlns:ex-schema=http://ex.org/schema#>
  <rdf:Description rdf:about="http://ex.org/ccf_adl">
     <ex-schema:speaker rdf:resource="http://ex.org/haofen"/>
     <ex-schema:theme rdf:resource="http://ex.org/KG"/>
   </rdf:Description>
</rdf:RDF>
```



- **Turtle**

```
@prefix ex: <http://ex.org/> .
@prefix ex-schema: <http://ex.org/schema#>
  ex:ccf_adl
  ex-schema:speaker ex:haofen;
  ex-schema:theme ex:KG.
```

- **N-Triples**

```
< http://ex.org/ccf_adl>
     <http://ex.org/schema#speaker>
     <http://ex.org/haofen>.

< http://ex.org/ccf_adl>
     <http://ex.org/schema#theme>
     <http://ex.org/KG>.
```

开放世界假设

与经典系统的封闭世界假设相对应

简而言之，一个三元组的缺失
并不是一件大不了的事情

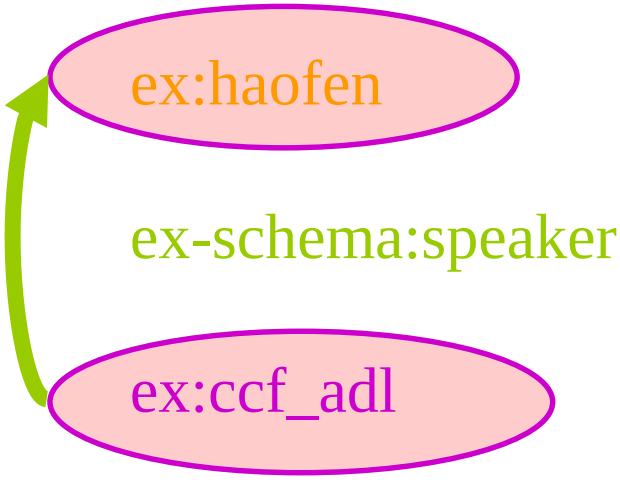
(CCF ADL, speaker, Haofen)

并不意味着CCF ADL讲座只有一位讲者

(CCF ADL, speaker, Haofen)

意味着CCF ADL讲座至少有一位讲者

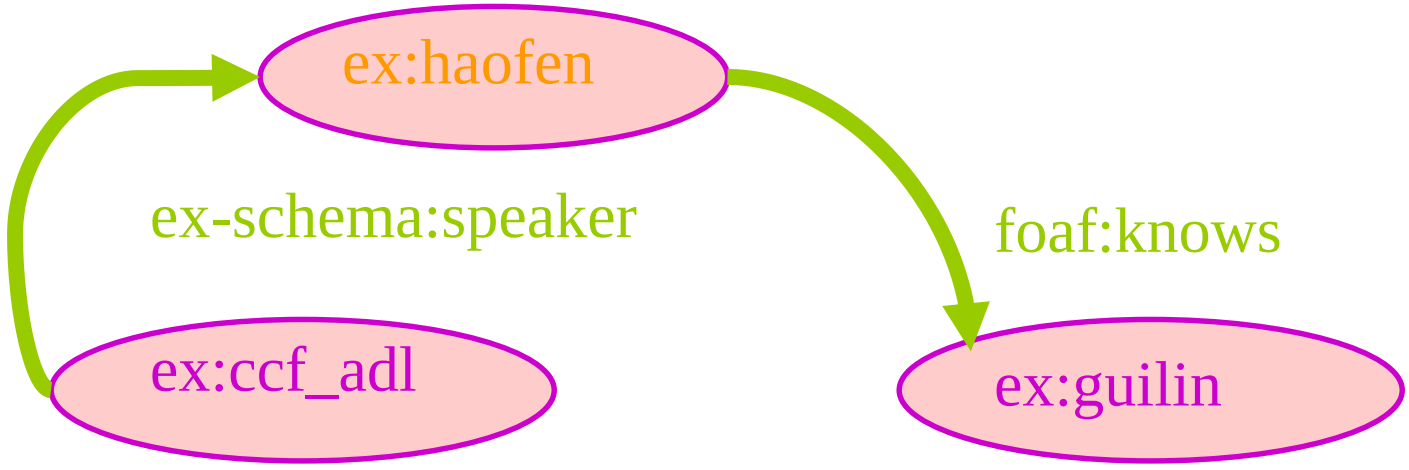
RDF允许分布式地定义知识



发布在CCF ADL讲座网页

在吴奋的主页中体现

分布式定义的知识可以自动合并



带标注RDF

- 如何扩展RDF用来表达更多信息
  - 时间、不确定性、空间、信任等
  - => 带标注 RDF(S)
  - 采用带标注RDF表示的知识库： YAGO2
- YAGO2：采用annotated RDF表示的知识库
- 语法形式：$(s, p, o): \lambda$
  - λ是一个标志
  - 例子（特朗普，就职，总统）：2017年1月

RDF和RDFS

RDF Schema (RDFS)

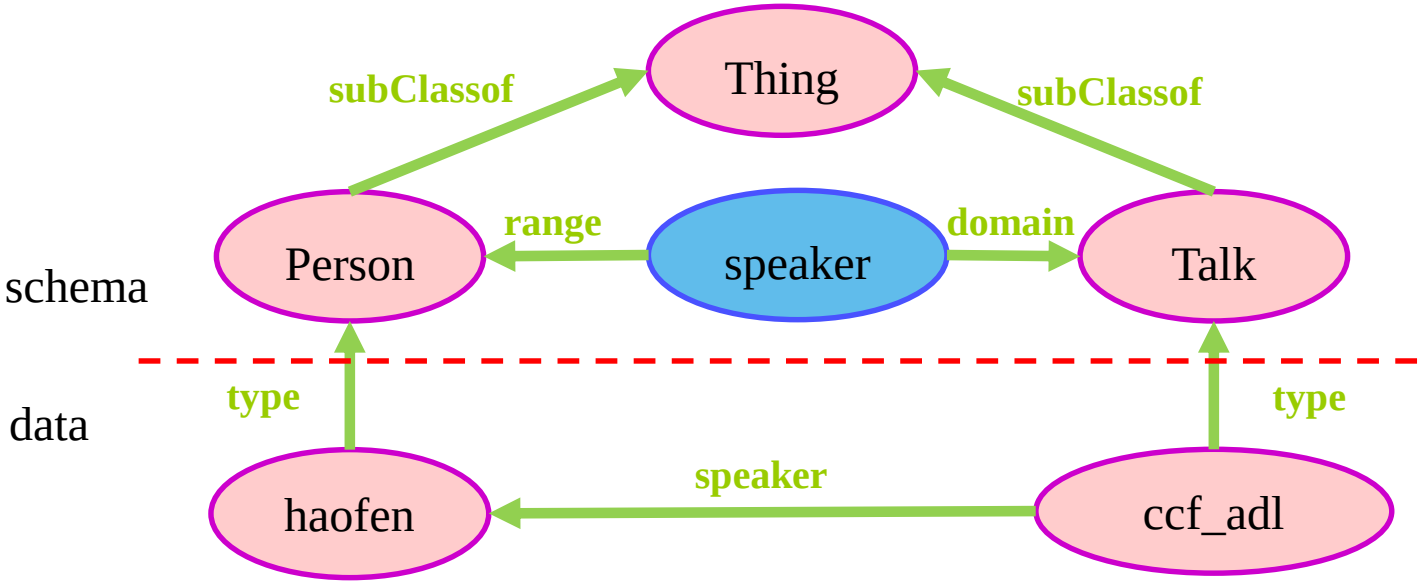
RDFS在RDF基础上提供了一个术语、概念等的定义方式，以及哪些属性可以应用到哪些对象上。换句话说，RDFS为RDF模型提供了一个基本的类型系统。

举例：
<http://mymetadata.vocab.org/Author>
<rdfs: subClassOf >
<http://purlorg/dc/elements/1.0/Creator>.

上述三元组表示用户自定义的元数据Author是Dublin Core的元数据Creator的子类。RDF Schema正是通过这样的方式来描述不同词汇集的元数据之间的关系，从而为网络上统一格式的元数据交换打下基础。

RDF Schema (RDFS)

为RDF定义了如下词汇:
Class, subClassOf, type, Property, subPropertyOf, Domain, Range



RDFS推理示例

谷歌 rdf:type 人工智能公司

人工智能公司 rdfs:subclass 高科技公司

谷歌 rdf:type 高科技公司

投资 rdfs:domain 投资人

投资 rdfs:range 公司

大卫·切瑞顿 投资 谷歌

大卫·切瑞顿 rdf:type 投资人

OWL和OWL2

RDF(S)表达能力上的缺陷

通过RDF(S)可以表示一些简单的语义，但在更复杂的场景下，RDF(S)语义表达能力显得太弱，还缺少诸多常用的特征。

对于局部值域的属性定义：RDF(S)中通过rdfs:range定义了属性的值域，该值域是全局性的，无法说明该属性应用于某些具体的类时具有的特殊值域限制。

类、属性、个体的等价性：RDF(S)中无法声明两个或多个类、属性和个体是等价还是不等价。

不相交类的定义： 在RDF(S)中只能声明子类关系，如男人和女人都是人的子类，但无法声明这两个类是不相交的。

OWL和OWL2

RDF(S)的缺陷

基数约束： 即对某属性值可能或必须的取值范围进行约束，如说明一个人有双亲（包括两个人），一门课至少有一名教师等。

关于属性特性的描述：即声明属性的某些特性，如传递性、函数性、对称性，以及声明一个属性是另一个属性的逆属性等。

W3C提出了OWL语言扩展RDF(S)，作为在语义网上表示本体的推荐语言

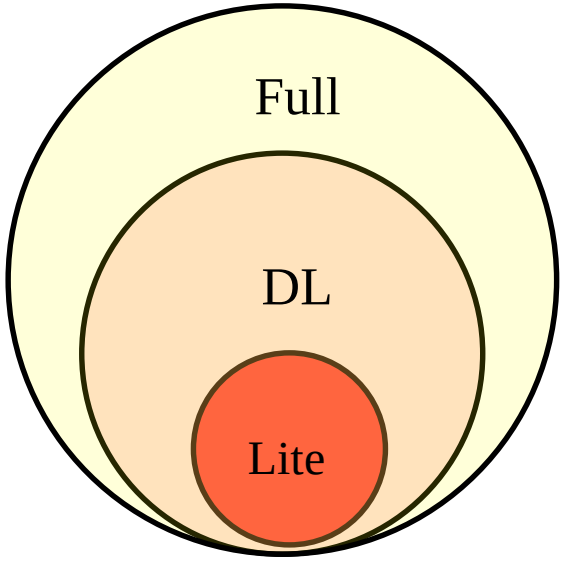
OWL和OWL2

OWL简介

W3C于2002年7月31日发布了OWL Web 本体语言 (OWL Web Ontology Language) 工作草案的细节，其目的是为了更好地开发语义网。

OWL的三个子语言

- OWL Lite
- OWL DL
- OWL Full



OWL和OWL2

OWL的三个子语言

子语言	特征	使用限制举例
OWL Lite	用于提供给那些只需要一个分类层次和简单的属性约束的用户。	支持基数(cardinality)，但允许基数为0或1。
OWL DL	在OWL Lite基础上包括了OWL语言的所有约束。该语言上的逻辑蕴涵是可判定的。	当一个类可以是多个类的一个子类时，它被约束不能是另外一个类的实例。
OWL Full	它允许在预定义的(RDF、OWL)词汇表上增加词汇，从而任何推理软件均不能支持OWL FULL的所有feature。OWL Full语言上的逻辑蕴涵通常是不可判定的。	一个类可以被同时表达为许多个体的一个集合以及这个集合中的一个个体。具有二阶逻辑特点。

OWL和OWL2

选择OWL子语言的考虑

- 选择OWL Lite还是OWL DL主要取决于用户需要整个语言在多大程度上给出约束的可表达性；
- 选择OWL DL还是OWL Full主要取决于用户在多大程度上需要RDF的元模型机制(如定义类型的类型以及为类型赋予属性)；
- 在使用OWL Full而不是OWL DL时，推理的支持可能不能工作，因为目前还没有完全的支持OWL Full的系统实现。

OWL和OWL2

OWL与RDF的关系

- OWL Full可以看成是RDF的扩展；
- OWL Lite和OWL Full可以看成是一个约束化的RDF的扩展；
- 所有的OWL文档 (Lite，DL，Full)都是一个RDF文档；
- 所有的RDF文档都是一个OWL Full文档；
- 只有一些RDF文档是一个合法的OWL Lite和OWL DL文档。

OWL和OWL2

OWL词汇

- 等价性声明

exp:运动员 owl:equivalentClass exp:体育选手

exp:获得 owl:equivalentProperty exp:取得

exp:运动员A owl:sameIndividualAs exp:小明

- exp是命名空间 http://www.example.org 的别称；
- 以上三个三元组分别声明了两个类，两个属性，以及两个个体是等价的。

OWL和OWL2

OWL词汇

- 声明属性的传递性

exp:ancestor rdf:type owl:TransitiveProperty

- exp:ancestor 是一个传递关系；
- 比如： exp:小明 exp:ancestor exp:小林； exp:小林 exp:ancestor exp:小志
  那么根据上述声明，有 exp:小明 exp:ancestor exp:小志 .

- 声明两个属性互反

exp:ancestor owl:inverseOf exp:descendant

- exp:ancestor和exp:descendant是互反关系；
- 比如： exp:小明 exp:ancestor exp:小林
  那么根据上述声明，有 exp:小林 exp:descendant exp:小明

OWL和OWL2

OWL词汇

- 声明属性的函数性

```
exp:hasMother rdf:type owl:FunctionalProperty
```

- exp:hasMother 是一个具有函数性的属性；因为每个人只有一个母亲。

- 声明属性的对称性

```
exp:friend rdf:type owl:SymmetricProperty
```

- exp:friend 是一个具有对称性的属性；
- 比如： exp:小明 exp:friend exp:小林
  那么根据上述声明，有 exp:小林 exp:friend exp:小明 .

OWL和OWL2

OWL词汇

- 声明属性的局部约束：全称限定

```
exp:Person owl:allValuesFrom exp:Women
exp:Person owl:onProperty exp:hasMother
```

- exp:hasMother在主语属于exp:Person类的时候，宾语的取值只能来自exp:Women这个类。

- 声明属性的局部约束：存在限定

```
exp:SemanticWebPaper owl:someValuesFrom exp:AAAI
exp:SemanticWebPaper owl:onProperty exp:publishedIn
```

- exp:publishedIn在主语属于exp:SemanticWebPaper类的时候，宾语的取值部分来自exp:AAAI这个类。

  上面的三元组相当于：关于语义网的论文部分发表在AAAI上。

OWL和OWL2

OWL词汇

- 声明属性的局部约束：基数限定

```
exp:Person owl:cardinality “1”^^xsd:integer
exp:Person owl:onProperty exp:hasMother
```

- exp:hasMother在主语属于exp:Person类的时候，宾语的取值只能有一个；
- “1”的数据类型被声明为xsd:integer；
- 这是基数约束，本质上属于属性的局部约束。

OWL和OWL2

OWL词汇

- 声明相交的类

```
exp:Mother owl:intersectionOf _tmp
_tmp rdf:type rdfs:Collection
_tmp rdfs:member exp:Person
_tmp rdfs:member exp:HasChildren
```

- _tmp是临时资源；它是rdfs:Collection类型，是一个容器；
  它的两个成员是exp:Person，exp:HasChildren；
- 上述三元组说明exp:Mother是exp:Person，exp:HasChildren这两个类的交集。

OWL和OWL2

OWL词汇扩展

OWL中的其它词汇	描述
owl:oneOf	声明枚举类型
owl:disjointWith	声明两个类不想交
owl:unionOf	声明类的并运算
owl:minCardinality owl:maxCardinality	最小最大的基数限定
owl:InverseFunctionalProperty	声明互反类具有函数属性
owl:hasValue	属性的局部约束时，声明所约束类必有一个取值

OWL和OWL2

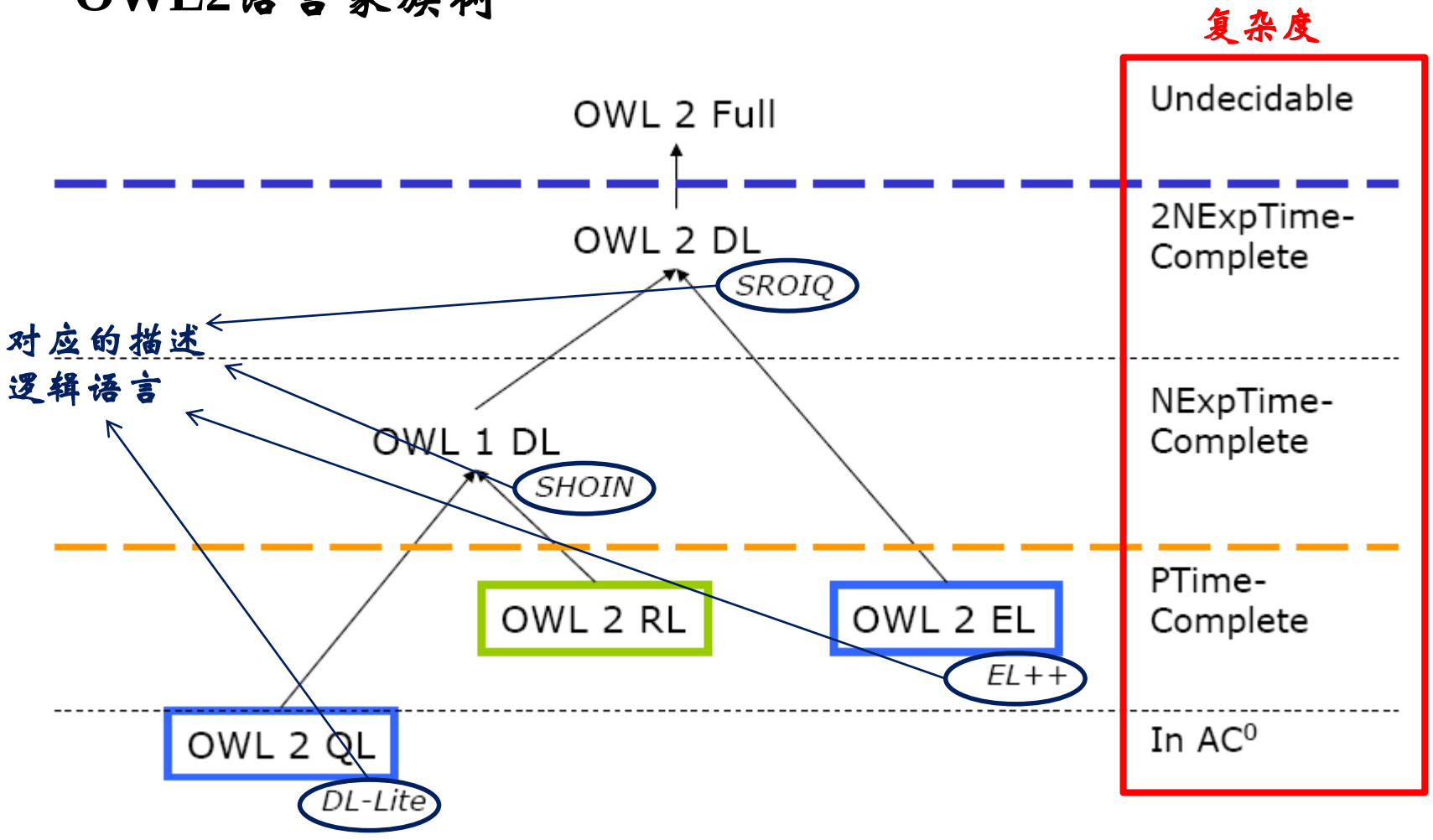
OWL2

- OWL的最新版本； 老的OWL版本也被称为OWL1；
- OWL2定义了一些OWL的子语言，通过限制语法使用，使得这些子语言能够更方便地实现，以及服务于不同的应用；
- OWL2的三大子语言：

  OWL 2 RL，OWL 2 QL，OWL 2 EL

OWL和OWL2

OWL2语言家族树



OWL和OWL2

OWL 2 QL

- QL代表query language的意思，专为基于本体的查询设计；
- OWL 2 QL的复杂度是AC^0，非常适合大规模处理；
- OWL 2的三大子语言中，QL最为简单；
- OWL 2 QL是基于描述逻辑语言DL-Lite定义的。

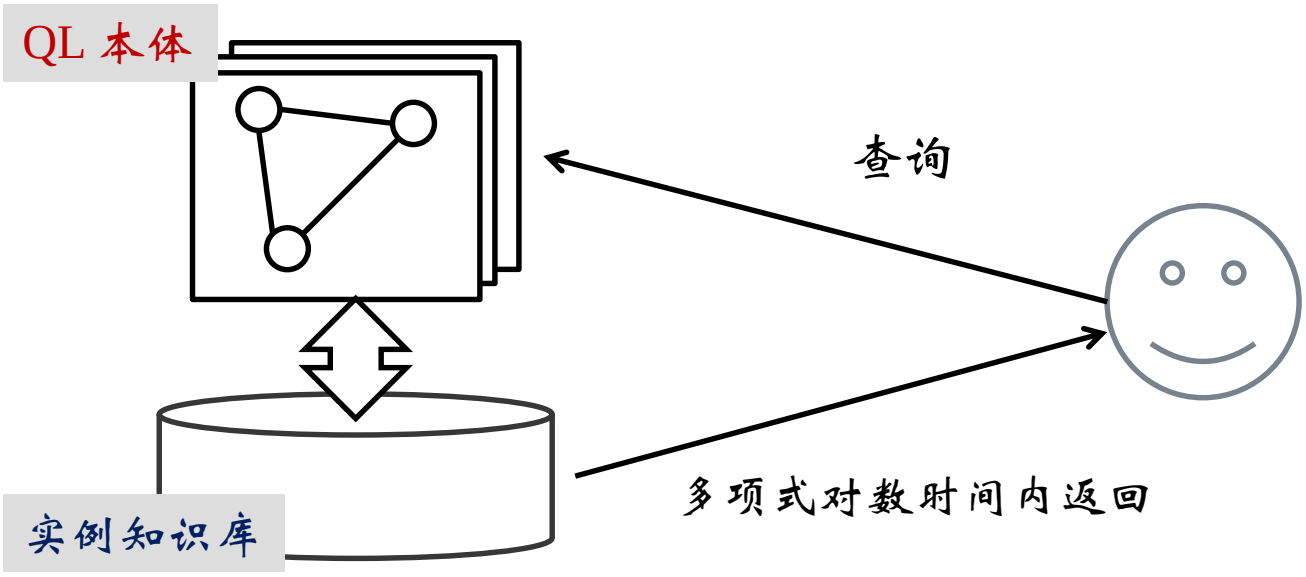
OWL和OWL2

OWL 2 QL 表达能力

允许的核心词汇	对应的描述逻辑公理举例
rdfs:subClassOf	$Mother \sqsubseteq Person$
rdfs:subPropertyOf	$hasSon \sqsubseteq hasChild$
rdfs:domain	$\exists hasSon.\top \sqsubseteq Person$
rdfs:range	$\top \sqsubseteq \forall hasSon.Person$
owl:inverseOf	$hasChild \equiv hasParent^{-}$
owl:disjointWith	$Women \sqcap Man \sqsubseteq \bot$

OWL和OWL2

OWL 2 QL 优势



通过OWL 2 QL的语言限制，基于QL的本体查询可以优化到**多项式对数**时间复杂度

OWL和OWL2

OWL 2 EL

- OWL 2 EL专为概念术语描述，推理而设计；
- 在生物医疗领域广泛应用，如临床医疗术语本体SNOMED CT；
- 复杂度是PTime-Complete；
- OWL 2 EL是基于描述逻辑语言 EL++定义的。

OWL和OWL2

OWL 2 EL 表达能力

允许的核心词汇	对应的描述逻辑公理举例
rdfs:subClassOf	Mother ⊑ Person
rdfs:subPropertyOf	hasSon ⊑ hasChild
owl:someValuesOf	∃hasSon.Children ⊑ Person Parent ⊑ ∃hasSon.Children
owl:intersectionOf	Star ⊓ Women ⊑ Scandal
owl:TransitiveProperty	Tran(hasAncestor)

OWL 2 EL 允许表达如下复杂的概念

Female ⊓ ∃likes.Movie ⊓ ∃hasSon.(Student ⊓ ∃attends.CSCourse)

所有喜欢电影，儿子是学生且参加计算机课程的女性

OWL和OWL2

OWL 2 EL推理的例子

$Apple \sqsubseteq \exists beInvestedBy.(Fidelity \sqcap BlackStone)$

苹果由富达和黑石投资。

$\exists beFundedBy.Fidelity \sqsubseteq InnovativeCompanies$

借助富达融资的公司都是创新企业。

$\exists beFundedBy.BlackStone \sqsubseteq InnovativeCompanies$

借助黑石融资的公司都是创新企业。

$beInvestedBy \sqsubseteq beFundedBy$

投资即是帮助融资。

OWL和OWL2

OWL 2 EL推理的例子

$\text{Apple} \sqsubseteq \exists \text{beInvestedBy}.(\text{Fidelity} \sqcap \text{BlackStone})$

$\exists \text{beFundedBy}.\text{Fidelity} \sqsubseteq \text{InnovativeCompanies}$

$\exists \text{beFundedBy}.\text{BlackStone} \sqsubseteq \text{InnovativeCompanies}$

$\text{beInvestedBy} \sqsubseteq \text{beFundedBy}$

OWL和OWL2

OWL 2 EL推理的例子

Apple ⊑ ∃beInvestedBy.(Fidelity ⊓BlackStone)

∃beFundedBy.Fidelity ⊑ InnovativeCompanies

∃beFundedBy.BlackStone ⊑ InnovativeCompanies

beInvestedBy ⊑ beFundedBy

Apple ⊑ ∃beInvestedBy.Fidelity

OWL和OWL2

OWL 2 EL推理的例子

$\text{Apple} \sqsubseteq \exists \text{beInvestedBy}.(\text{Fidelity} \sqcap \text{BlackStone})$

$\exists \text{beFundedBy}.\text{Fidelity} \sqsubseteq \text{InnovativeCompanies}$

$\exists \text{beFundedBy}.\text{BlackStone} \sqsubseteq \text{InnovativeCompanies}$

$\text{beInvestedBy} \sqsubseteq \text{beFundedBy}$

$\text{Apple} \sqsubseteq \exists \text{beInvestedBy}.\text{Fidelity}$

$\text{Apple} \sqsubseteq \exists \text{beFundedBy}.\text{Fidelity}$

OWL和OWL2

OWL 2 EL推理的例子

Apple ⊑ ∃beInvestedBy.(Fidelity ⊓BlackStone)

∃beFundedBy.Fidelity ⊑ InnovativeCompanies

∃beFundedBy.BlackStone ⊑ InnovativeCompanies

beInvestedBy ⊑ beFundedBy

Apple ⊑ ∃beInvestedBy.Fidelity

Apple ⊑ ∃beFundedBy.Fidelity

Apple ⊑ InnovativeCompanies

OWL和OWL2

OWL 2 RL

- OWL 2 RL在扩展RDFS表达能力的同时，保持了较低的复杂度；
- OWL 2 RL在RDFS的基础上引入属性的特殊特性(函数性，互反性，对称性)；允许声明等价性；允许属性的局部约束；
- 复杂度是PTime-Complete；

OWL和OWL2

OWL 2 RL 设计初衷

- OWL 2 RL在ter Horst的工作基础[1]上延伸而来；该工作的目的是将OWL词汇引入RDFS，使得RDFS在表达能力上丰富起来，同时保持计算复杂度在PTime级别；
- OWL 2 RL与描述逻辑没有直接关系；
- 业界的一种观点是，OWL 2 RL是专为高效推理设计的本体语言。

[1] ter Horst, H. J. Completeness, decidability and complexity of entailment for RDF Schema and a semantic extension involving the OWL vocabulary. *J. Web Sem.*, **2005**, *3*, 79-115.

OWL和OWL2

OWL 2 RL 表达能力

允许的核心词汇
rdfs:subClassOf
rdfs:subPropertyOf
rdfs:domain
rdfs:range
owl:TransitiveProperty
owl:FunctionalProperty
owl:sameAs
owl:equivalentClass
owl:equivalentProperty
owl:someValuesFrom
owl:allValuesFrom

OWL和OWL2

OWL 2 RL 推理简介

- OWL 2 RL推理是针对于**实例数据**的推理；PTime复杂度也是针对实例数据推理得到的结果；

- 针对定义域值域的推理

p rdfs:domain x, s p o ⇒ s rdf:type x
p rdfs:range x, s p o ⇒ o rdf:type x

- s, p, o, x为变量；
- 例子：由 exp:hasChild rdfs:domain exp:Person,
  exp:Helen exp:hasChild exp:Jack，
  有 exp:Helen rdf:type exp:Person。

OWL和OWL2

OWL 2 RL 推理简介

- subClassOf和subPropertyOf的传递性

p rdfs:subPropertyOf q, q rdfs:subPropertyOf r ⇒
p rdfs:subPropertyOf r

v rdfs:subClassOf w, w rdfs:subClassOf x ⇒
v rdfs:subClassOf x

- p, q, r, v, w, x为变量；
- 例子：由 exp:hasSon rdfs:subPropertyOf exp:hasChildren,
  exp:hasChildren rdfs:subPropertyOf exp:hasOffspring，
  有 exp:hasSon rdfs:subPropertyOf exp:hasOffspring。

OWL和OWL2

OWL 2 RL 推理简介

- 针对自定义传递性属性推理

p rdf:type owl:TransitiveProperty, u p w, w p v ⇒ u p v

- 针对互反属性推理

p owl:inverseOf q, v p w ⇒ w q v
p owl:inverseOf q, v q w ⇒ w p v

OWL和OWL2

OWL 2 RL 推理简介

- 针对等价性推理

v owl:sameAs w, w owl:sameAs u ⇒ v owl:sameAs u

v owl:equivalentClass w ⇒ v rdfs:subClassOf w

v owl:equivalentProperty w ⇒ v rdfs:subPropertyOf w

v rdfs:subPropertyOf w, w rdfs:subPropertyOf v ⇒
v rdfs:equivalentProperty w

v rdfs:subClassOf w, w rdfs:subClassOf v ⇒
v rdfs:equivalentClass w

OWL和OWL2

OWL 2 RL 推理简介

- 针对属性局部约束的推理

> v owl:allValuesFrom u, v owl:onProperty p,
> w rdf:type v, w p x ⇒ x rdf:type u

- 例子：由 exp:SWPaper owl:allValuesFrom exp:AAAI,
  exp:SWPaper owl:onProperty exp:publishedIn，
  exp:paper1 rdf:type exp:SWPaper,
  exp:paper1 exp:publishedIn exp:conference1,
  有 exp:conference1 rdf:type exp:AAAI。

OWL和OWL2

OWL2现有的推理系统

- OWL 2 DL reasoners
  FaCT++ (Manchester), HermiT (Oxford), Pellet (Clarkparsia)
- OWL 2 EL reasoners
  CEL (Dresden), ELK (Oxford)
- OWL 2 RL reasoners
  Jena (HP Labs Bristol, Aberdeen), Oracle 11g OWL Reasoner (Oracle)
- OWL 2 QL reasoners
  QuOnto (Rome), Quill (Aberdeen)

SPARQL简介

- RDF的查询语言
  - 基于RDF数据模型
- 可以对不同的数据集撰写复杂的连接(joins)
- 由所有主流图数据库支持
- **S**PARQL **P**rotocol and **R**DF **Q**uery **L**anguage

详见：http://www.w3.org/TR/rdf-sparql-query/

SPARQL查询结构

A SPARQL query comprises, in order:

- *Prefix declarations*, for abbreviating URIs
- *Dataset definition*, stating what RDF graph(s) are being queried
- A *result clause*, identifying what information to return from the query
- The *query pattern*, specifying what to query for in the underlying dataset
- *Query modifiers*, slicing, ordering, and otherwise rearranging query results

```
# prefix declarations
PREFIX foo: <http://example.com/resources/>
...
# dataset definition
FROM ...
# result
                    clause
SELECT ...
# query pattern
WHERE {
    ...
}
# query modifiers
ORDER BY ...
```

SPARQL查询基本构成

- 变量，RDF中的资源，以“?”或者“$”指示；
- 三元组模板（triple pattern），在WHERE子句中列示关联的三元组模板，之所以称之为模板，因为三元组中允许变量；
- SELECT子句中指示要查询的目标变量。

SPARQL查询语言

一个简单的SPARQL查询

```
PREFIX exp: http://www.example.org/
SELECT ?student
WHERE {
   ?student exp:studies exp:CS328.
 }
```

- 查询所有选修CS328课程的学生；
- 和数据库的SQL语言进行对应；
- PREFIX部分进行命名空间的声明，使得下面查询的书写更为简洁。

SPARQL查询语言

SPARQL查询其它关键字简介

- OPTIONAL

```
SELECT ?student  ?email
WHERE {
    ?student exp:studies exp:CS328 .
OPTIONAL {
    ?student foaf:mbox ?email .
   }
}
```

- 查询所有选修CS328课程的学生姓名，以及他们的邮箱；OPTIONAL关键字指示如果没有邮箱，那么依然返回学生姓名，邮箱处空缺。

SPARQL查询语言

SPARQL查询其它关键字简介

- FILTER

```
SELECT ?module ?name ?age
WHERE {
    ?student exp:studies ?module .
    ?student foaf:name ?name .
OPTIONAL {
    ?student exp:age ?age .
FILTER (?age > 25) }
 }
```

- 查询学生姓名，选修课程，以及他们的年龄；如果有年龄，那么年龄必须大于25岁。

SPARQL查询语言

SPARQL查询其它关键字简介

- UNION

```
SELECT ?student ?email
WHERE {
      ?student foaf:mbox ?email .
     { ?student exp:studies exp:CS328 }
     UNION { ?student exp:studies exp:CS909 }
}
```

- 查询选修课程CS328或CS909的学生姓名以及邮件；
- 注意，这里的邮件是必须返回的，如果没有邮件值，那么就不返回这条记录；注意和OPTIONAL区别。

SPARQL查询语言

SPARQL查询其它关键字简介

- FROM

```
SELECT ?student ?email  ?home
FROM <http://www2.warwick.ac.uk/rdf/student>
WHERE {
 ?student exp:studies exp:CS909 .
OPTIONAL { ?student foaf:mbox?email .
                ?student foaf:homepage ?home. }
}
```

- 查询选修课程CS909的学生姓名以及邮件和住址；
- FROM关键字引入了其它本体或者可访问的知识库。

SPARQL查询语言

SPARQL查询举例：

finance:融创中国 rdf:type finance:地产事业
finance:孙宏斌 finance:control finance:融创中国
finance:贾跃亭 finance:control finance:乐视网
finance:孙宏斌 finance:hold_share finance:乐视网
finance:王健林 finance:control finance:万达集团
finance:万达集团 finance:main_income finance:地产事业
finance:融创中国 finance:acquire finance:乐视网
finance:融创中国 finance:acquire finance:万达集团

```
SELECT ?P ?X
WHERE {
        ?P finance:control ?c .
        ?c finance:acquire ?X .
}
```

查询所有的收购关系。

SPARQL查询语言

SPARQL查询举例:

finance:融创中国 rdf:type finance:地产事业
finance:孙宏斌 finance:control finance:融创中国
finance:贾跃亭 finance:control finance:乐视网
finance:孙宏斌 finance:hold_share finance:乐视网
finance:王健林 finance:control finance:万达集团
finance:万达集团 finance:main_income finance:地产事业
finance:融创中国 finance:acquire finance:乐视网
finance:融创中国 finance:acquire finance:万达集团

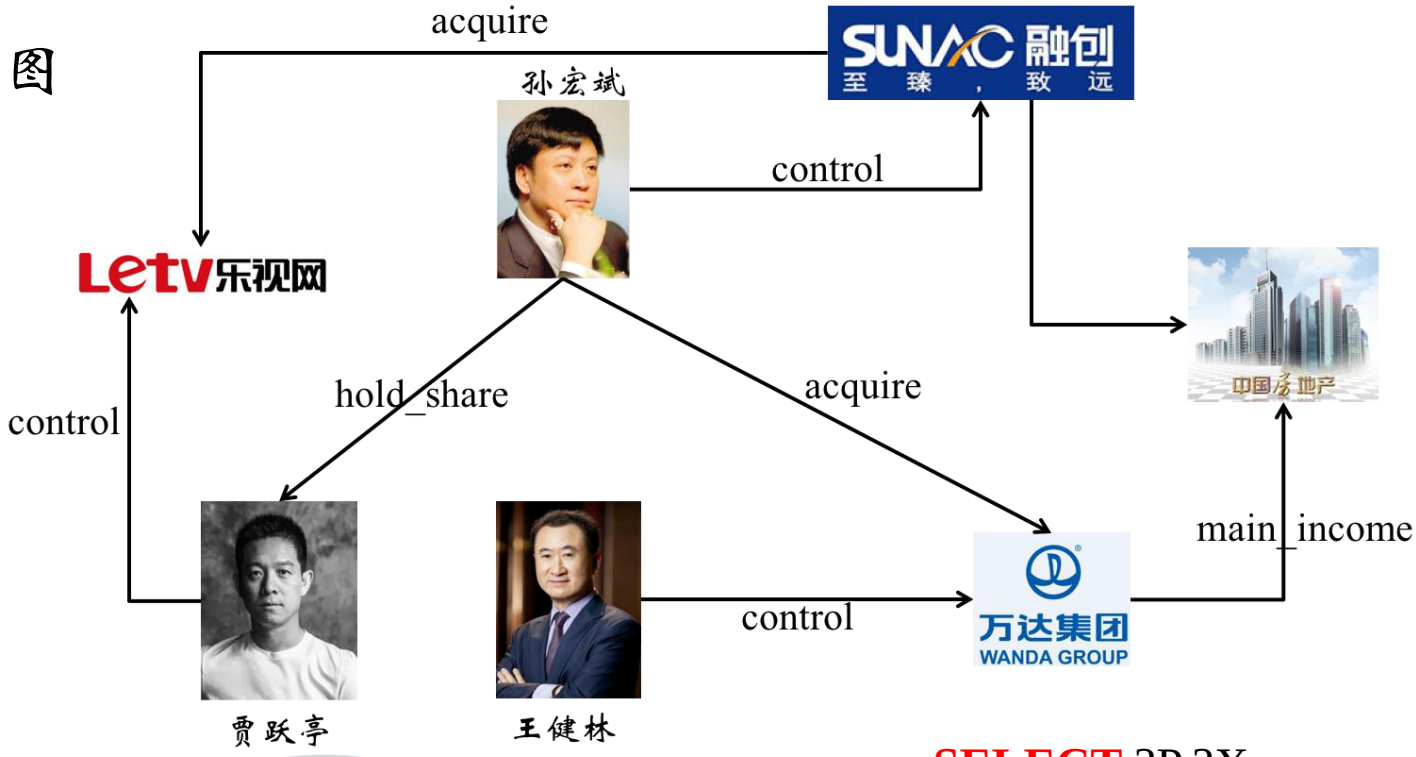
```
SELECT ?P ?X
WHERE {
        ?P finance:control ?c .
        ?c finance:acquire ?X .
}
```

查询结果

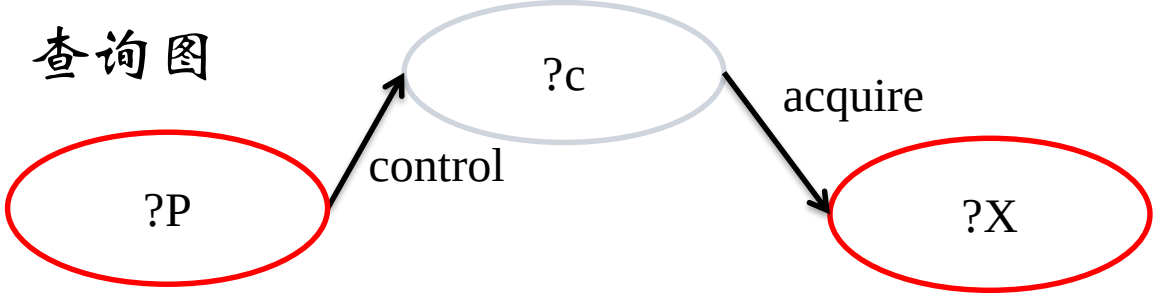
?P	?X
孙宏斌	乐视网
孙宏斌	万达集团

SPARQL查询图

RDF数据图



查询图




```
SELECT ?P ?X
WHERE {
        ?P finance:control ?c .
        ?c finance:acquire ?X .
}
```

更多SPARQL查询示例

```
PREFIX foaf:   <http://xmlns.com/foaf/0.1/>
PREFIX org:    <http://example.com/ns#>
CONSTRUCT { ?x foaf:name ?name }
WHERE  { ?x org:employeeName ?name }
```

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
ASK    { ?x foaf:name  "Alice" }
```

"Find the people who live in "Palo Alto" and have founded or are board members of companies in the software industry. For each such company, find the products that were developed by it, its revenue, and optionally its number of employees."

```
SELECT* WHERE
{ ?x home "Palo Alto" .
      { ?x founder ?y } UNION { ?x member ?y }
      {
            ?y industry "Software" .
            ?z developer ?y .
            ?y revenue ?n .
            OPTIONAL { ?y employees ?m } .
      }
}
```

SPARQL 1.1:
Everything you loved about SPARQL plus Aggregates, Sub-queries, Property paths, Negation and more!

RDF提供了知识建模的灵活性，但查询可能会成为噩梦

A very simple query to express at the conceptual level …

> "Find people who are members of an organization founded by a relative."

but tedious to formulate precisely against RDF data

```
SELECT* WHERE
{ { ?x member ?y } UNION
      { ?x board_member ?y } UNION
      { ?x ceo ?y } UNION
      { ?x works_for ?y } UNION
      { ?x employee ?y } UNION
      { ?y employer ?x } UNION
      { ?x intern ?y } UNION ….
      { ?z founder ?y }
      { ?z brother ?x } UNION
      { ?x brother ?z } UNION
      { ?z father ?x } UNION
      { ?x father ?z } UNION
      { ?z mother ?x } UNION
      { ?x mother ?z } UNION …
}
```

Every possible way membership in an organization can be expressed in the RDF data

Every possible way family relationship can be expressed in the RDF data

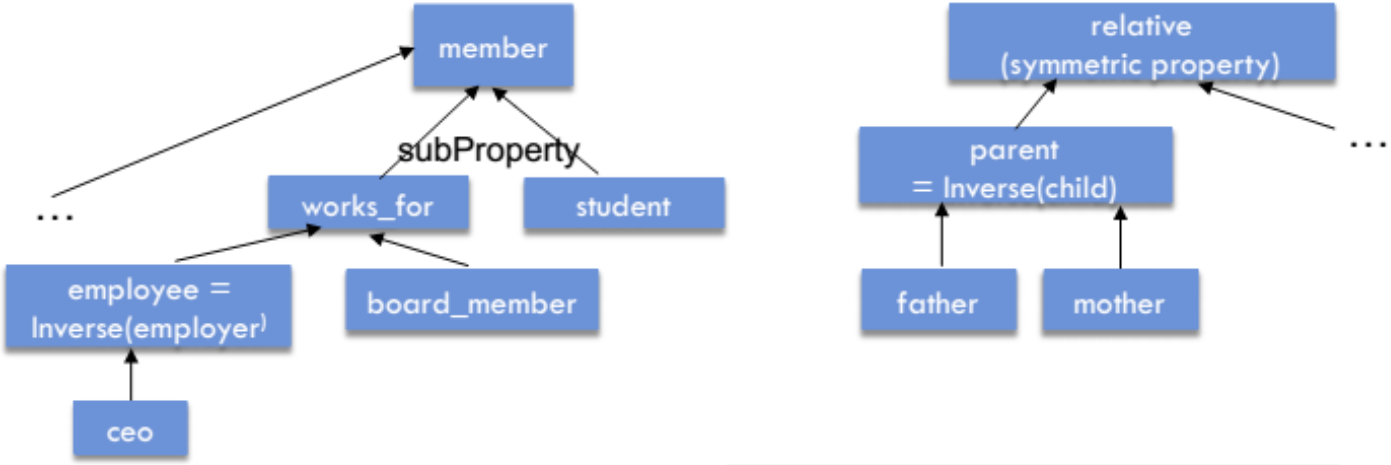
RDF provides the flexibility to use any predicate and any type in any statement but no mechanism to define the meaning of those predicates and types

本体 (ontology)可以填充知识与查询之间的语义间隙

Let's revisit our simple conceptual query

"Find people who are members of an organization founded by a relative."

Assume some mechanism to specify relationships between predicates as follows:



Simpler precise query against the RDF data enriched with the above semantic knowledge (ontology)

```
SELECT* WHERE {
  ?x member ?y .
  ?z founder ?y .
  ?z relative ?x
}
```

SPARQL查询语言

SPARQL查询举例:

finance:融创中国 rdf:type finance:地产事业
finance:孙宏斌 finance:control finance:融创中国
finance:贾跃亭 finance:control finance:乐视网
finance:孙宏斌 finance:hold_share finance:乐视网
finance:王健林 finance:control finance:万达集团
finance:万达集团 finance:main_income finance:地产事业
finance:融创中国 finance:acquire finance:乐视网
finance:融创中国 finance:acquire finance:万达集团

```
SELECT ?X ?Y
WHERE {
        ?X finance:conn_trans ?Y .
}
```

查询关联交易的公司

SPARQL查询语言

SPARQL查询举例：

写成规则的形式：

hold_share(X, Y) : - control(X, Y)

conn_trans(Y,Z) : - hold_share(X, Y), hold_share(X, Z)

```
SELECT ?X ?Y
WHERE {
        {?Z finance:control ?X .
       ?Z finance:control ?Y. }
   UNION {?Z finance:hold_share ?X.
          ?Z finance:hold_share ?Y. }
   UNION{?Z finance:control ?X .
          ?Z finance:hold_share ?Y.}
    UNION{?Z finance:hold_share ?X.
          ?Z finance:control ?Y.}
}
```

有没有更简单的方式?

SPARQL查询语言

SPARQL查询举例:

写成规则的形式:

hold_share(X, Y) : - control(X, Y)
conn_trans(Y,Z) : - hold_share(X, Y), hold_share(X, Z)

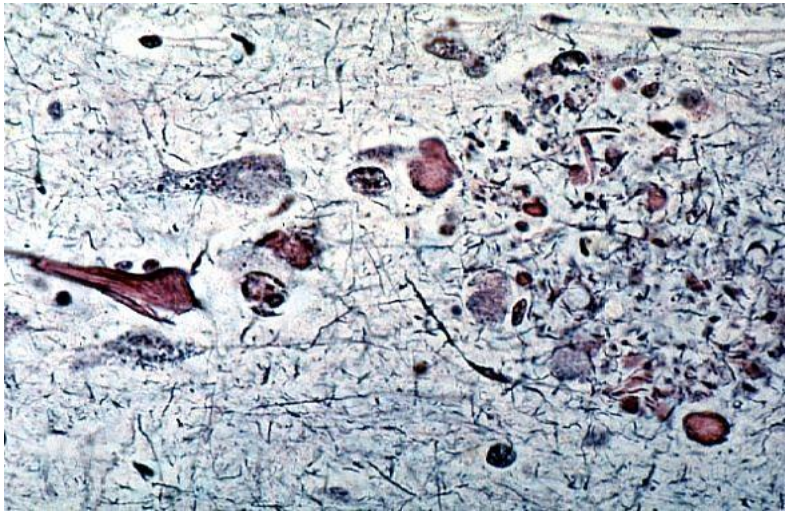
```
SELECT DISTINCT ?X ?Y
WHERE {
    {SELECT ?U ?X WHERE{?U finance:hold_share ?X .}}
    {SELECT ?U ?Y WHERE{?U finance:control ?Y .}}
}
```

- SPARQL允许嵌套查询，即WHERE子句中包含SELECT子句。

SPARQL用于新药发现的案例

查找阿尔茨海默病的靶标

- 信号转导通路(Signal transduction pathways)在药物靶标中很丰富——对化学疗法有反应的蛋白
- CA1锥体神经元(CA1 Pyramidal Neurons) 在阿尔茨海默病中遭到了损伤
- 我们可以找到参与信号转导和活跃在锥体神经元中的候选基因(candidate genes)吗?



跨知识库SPARQL查询示例

A SPARQL Query Spanning 4 Sources

```
prefix go: <http://purl.org/obo/owl/GO#>
prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
prefix owl: <http://www.w3.org/2002/07/owl#>
prefix mesh: <http://purl.org/commons/record/mesh/>
prefix sc: <http://purl.org/science/owl/sciencecommons/>
prefix ro: <http://www.obofoundry.org/ro/ro.owl#>

select ?genename ?processname
where
{ graph <http://purl.org/commons/hcls/pubmesh>
   { ?paper ?p mesh:D017966 .
     ?article sc:identified_by_pmid ?paper.
     ?gene sc:describes_gene_or_gene_product_mentioned_by ?article.
   }
  graph <http://purl.org/commons/hcls/goa>
   { ?protein rdfs:subClassOf ?res.
     ?res owl:onProperty ro:has_function.
     ?res owl:someValuesFrom ?res2.
     ?res2 owl:onProperty ro:realized_as.
     ?res2 owl:someValuesFrom ?process.
  graph <http://purl.org/commons/hcls/20070416/classrelations>
   {{?process <http://purl.org/obo/owl/obo#part_of> go:GO_0007166}
     union
    {?process rdfs:subClassOf go:GO_0007166 }}
     ?protein rdfs:subClassOf ?parent.
     ?parent owl:equivalentClass ?res3.
     ?res3 owl:hasValue ?gene.
   }
  graph <http://purl.org/commons/hcls/gene>
   { ?gene rdfs:label ?genename }
  graph <http://purl.org/commons/hcls/20070416>
   { ?process rdfs:label ?processname}
}
```

Mesh: Pyramidal Neurons

↓

Pubmed: Journal Articles

↓

Entrez Gene: Genes

↓

GO: Signal Transduction

Inference required

DRD1, 1812	adenylate cyclase activation
ADRB2, 154	adenylate cyclase activation
ADRB2, 154	arrestin mediated desensitization of G-protein coupled receptor protein signaling pathway
DRD1IP, 50632	dopamine receptor signaling pathway
	ase activating pathway
	ase inhibiting pathway
	signaling pathway
	signaling pathway
	signaling pathway
	signaling pathway
	signaling pathway
	signaling pathway

SPARQL makes ad hoc queries over multiple data sources (in RDF) easy

JSON-LD

- JSON-LD是JavaScript Object Notation for Linked Data的缩写，是一种基于JSON表示和传输互联数据 (Linked Data)的方法。JSON-LD描述了如何通过JSON表示有向图，以及如何在一个文档中混合表示互联数据及非互联数据。JSON-LD的语法和JSON兼容

- JSON-LD处理算法和API (JSON-LD Processing Algorithms and API)描述了处理JSON-LD数据所需的算法及编程接口，通过这些接口可以在JavaScript, Python及Ruby等编程环境中直接对JSON-LD文档进行转换和处理

JSON-LD

一个简单的JSON文本

```
{
"name": "Manu Sporny",
"homepage": "http://manu.sporny.org/",
"image": "http://manu.sporny.org/images/manu.png"
 }
```

- 这个JSON文档表示一个人。人们很容易推断这里的含义：“name”是人的名字，“homepage”是其主页，“image”是其某种照片。然而机器不理解“name”和“image”这样的术语。

JSON-LD

JSON-LD在做什么？

```
{
"http://schema.org/name": "Manu Sporny",
"http://schema.org/url": { "@id":"http://manu.sporny.org/" }, "http://schema.org/image":
{ "@id":"http://manu.sporny.org/images/manu.png" }
}
```

- JSON-LD通过引入规范的术语表示，比如统一化表示“name”，“homepage”和“image”的URI，使得数据交换和机器理解成为基础。

JSON-LD

JSON-LD和语义网

- JSON-LD呈现出语义网技术的风格，它们有着类似的目标：围绕某类知识提供共享的术语。例如，每个数据集不应该围绕“name”重复发明概念。
- JSON-LD的实现没有选择大部分语义网技术栈 (Turtle/SPARQL/Quad Stores)，而是以简单、不复杂以及面向一般开发人员的方式推进。

RDFa

- RDFa (Resource Description Framework in attributes)是网页标记语言
- RDFa是W3C推荐标准。它扩充了XHTML的几个属性，网页制作者可以利用这些属性在网页中添加可供机器读取的资源
- 与RDF的对应关系使得RDFa可以将RDF的三元组嵌入在XHTML文档中，它也使得符合标准的使用端可以从RDFa文件中提取出这些RDF三元组来

RDFa工作原理

- 通过引入名字空间的方法在已有的标签中加入RDFa相应的属性来使得支持RDFa技术的浏览器或者搜索引擎可以解析到，从而达到优化的目的。

```
<div xmlns:dc="http://purl.org/dc/elements/1.1/"
about="http://www.example.com/books/wikinomics">
   <span property="dc:title">Wikinomics</span>
   <span property="dc:creator">Mr right</span>
   <span property="dc:date">2006-09-02</span>
</div>
```

- 上面的代码示例中用到了RDFa属性中的about属性和property属性，这段代码示例说明了一篇文章，然后描述了和这篇文章相关的信息，比如说标题，创建者和创建日期，而这些属性就可以使得支持RDFa的机器识别。

从机器可理解的层面优化搜索，提升访问性以及网页数据的关联性。

HTML5 Microdata

- Microdata微数据，是在网页标记标记语言嵌入机器可读的属性数据
- 微数据使用可以来自自定义词汇表、带作用域的键/值对给DOM做标记
- 用户可以自定义微数据词汇表，在自己的网页中嵌入自定义的属性
- 微数据是给那些已经在页面上可见的数据施加额外的语义。当HTML的词汇不够用时，使用微数据可以取得较好的效果

HTML5 Microdata

定义一个概念类型
通过itemtype指示

```
<section itemscope itemtype="http://data-vocabulary.org/Person">
<h1 itemprop="name">Andy</h1>
<p><img itemprop="photo" src="http://www.example.com/photo.jpg"></p>
<a itemprop="url" href="http://www.example.com/blog">My Blog</a>
</section>
```

实体Andy的两个属性，
通过itemprop标签指示

在自定义命名空间
下的实体Andy

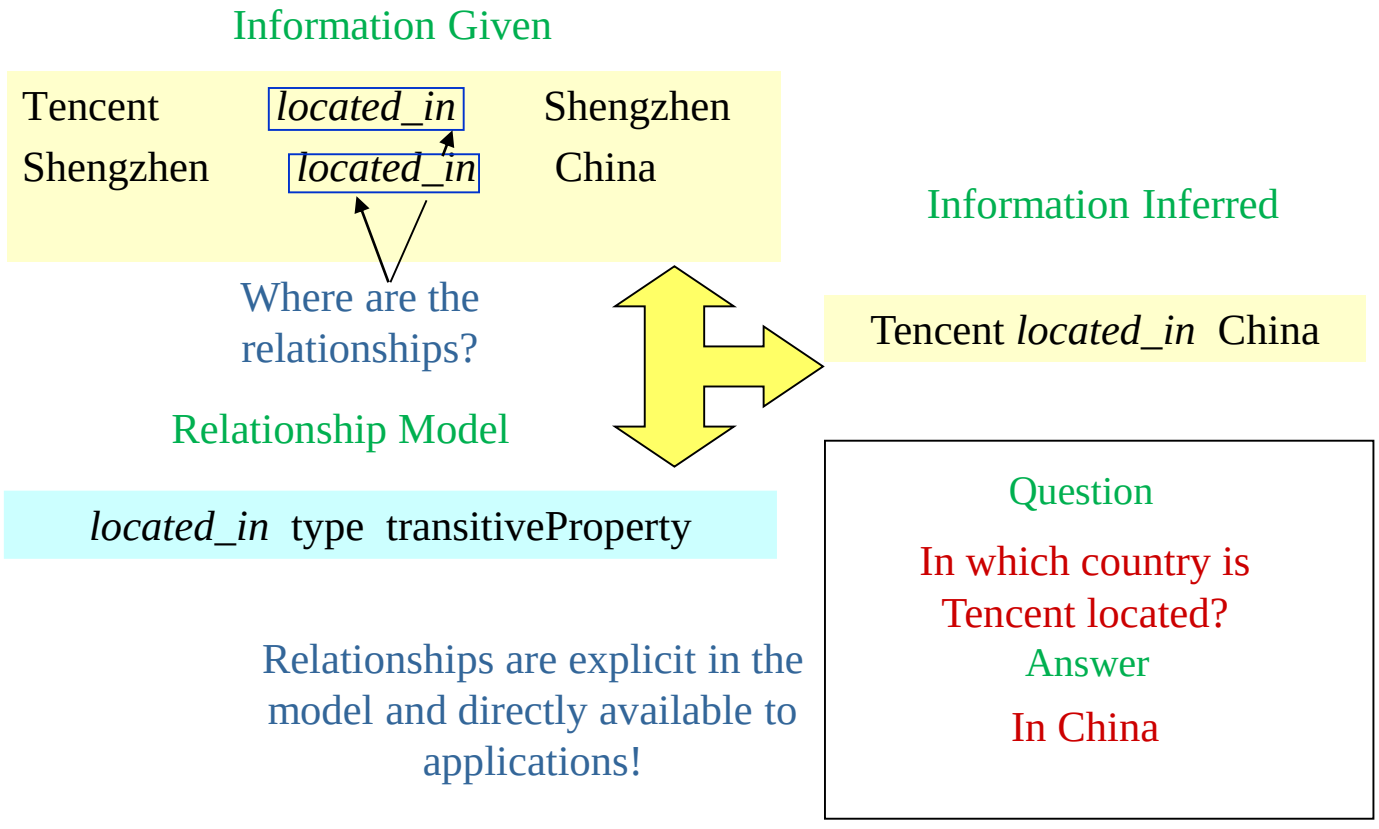
浏览器可以很方便地从网页上提取微数据实体、属性以及属性值。

RDF+SPARQL

对比

ER+SQL

RDF语义模型 – 关系显式定义



关系模型 – 关系隐式声明

Company Table

ID	Company Name
IDC	Tencent

Relationships are in documents, SQL code and collective memories - not available to applications!

City Table

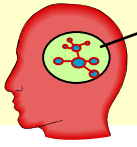
City	Country
Shengzhen	China

Company_City Table

City_ID	CO_ID
Shengzhen	IDC

Where are the relationships?

Data Definition Statements? Applications do not use them, they are not descriptive and their scope is a single database



Pous Medulla

Data Dictionary? Data Registry? They are for human, not computer use

Question

In which country is Tencent located?

Develop a Query

Select Country

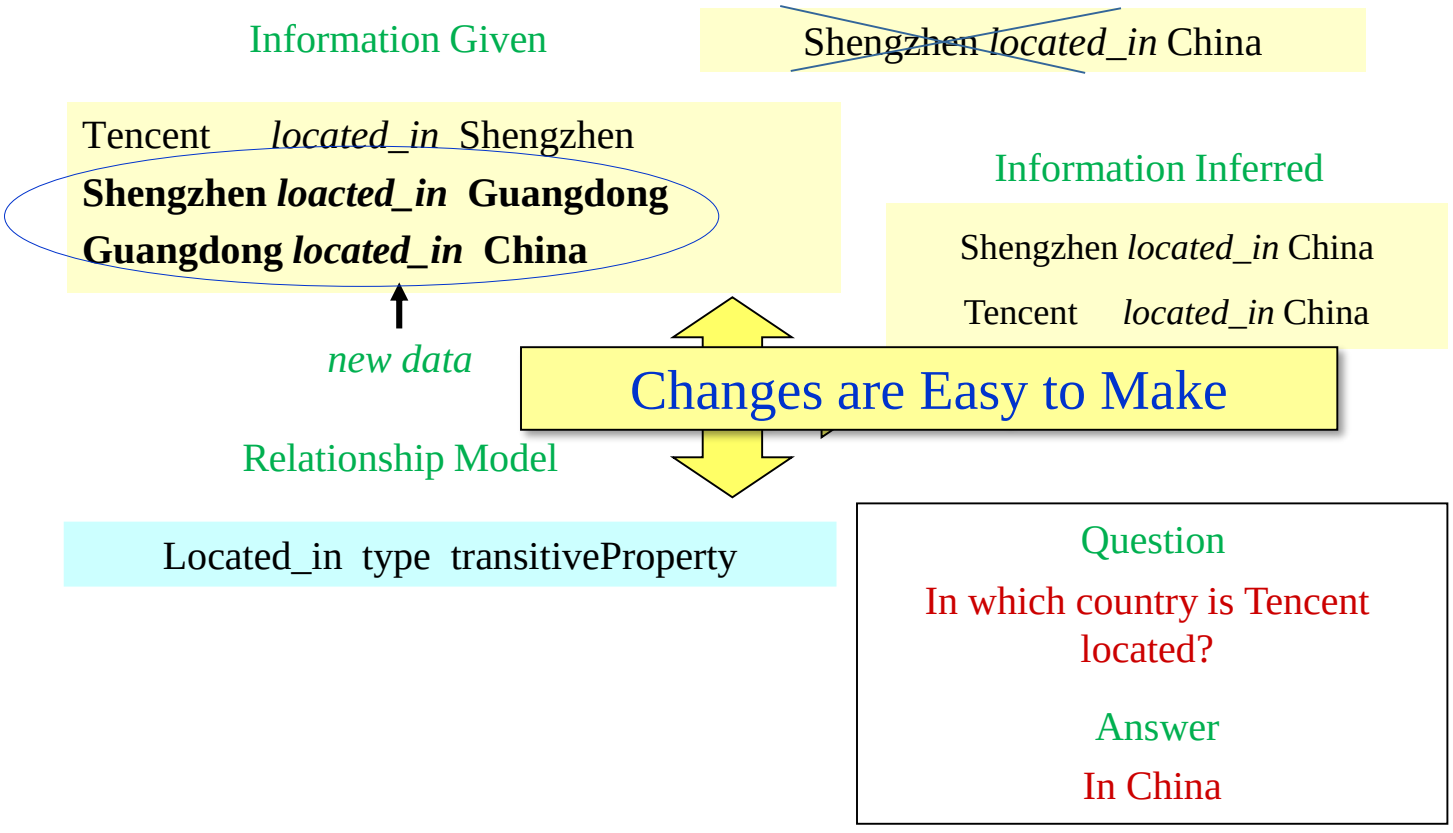
From Company Table, City Table, Company_City Table

Where Company Name = "Tencent" and ID = CO_ID and City = City_ID

Answer

In China

当数据发生了变更 – 语义模型如何应对？



当数据发生了变更 – 关系模型如何应对？

Company Table

ID	Company Name
IDC	Tencent

State Table

State_Name	ID	Country
Guangdong	GD	China

Company_City Table

City_ID	CO_ID
Shengzhen	IDC

City_State Table

City_ID	State_ID
Shengzhen	GD

Doesn't work any more!

Question

In which country is Tencent located?

Using the same query

Select Country

From Company Table, City Table, Company_City Table

Where Company Name = "Tencent" and ID = CO_ID and City = City_ID

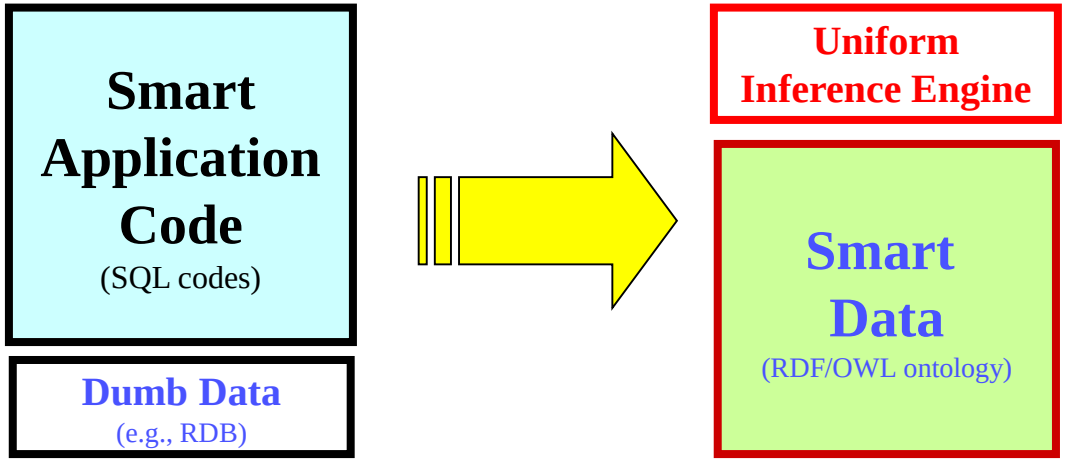
?

Get No Answer!

Changes should be avoided at ALL costs

数据的智能性如何体现？

知识建模和智能化服务构建的变革



第三部分：典型知识库项目的知识表示

各种知识图谱项目

PKUBASE

NELL

WEB CHILD

WEBKB

DBpedia抽取知识示例

About: 美国

http://dbpedia.org/page/United_States

An Entity of Type : bebouwde omgeving, from Named Graph : http://dbpedia.org, within Data Space : dbpedia.org

美利堅合眾國（英语：United States of America，簡稱：United States、America、The States，縮寫：U.S.A.、U.S.），簡稱為美國，是由50個州、華盛頓哥倫比亞特區、五个自治领土和外岛共同組成的聯邦共和国。美國本土48州和联邦特区位於北美洲中部，東臨大西洋，西臨太平洋，北面是加拿大，南部和墨西哥及墨西哥灣接壤，本土位於溫帶、副熱帶地區。阿拉斯加州位於北美大陸西北方，東部為加拿大，西隔白令海峽和俄羅斯相望；夏威夷州則是太平洋中部的群島。美國在加勒比海和

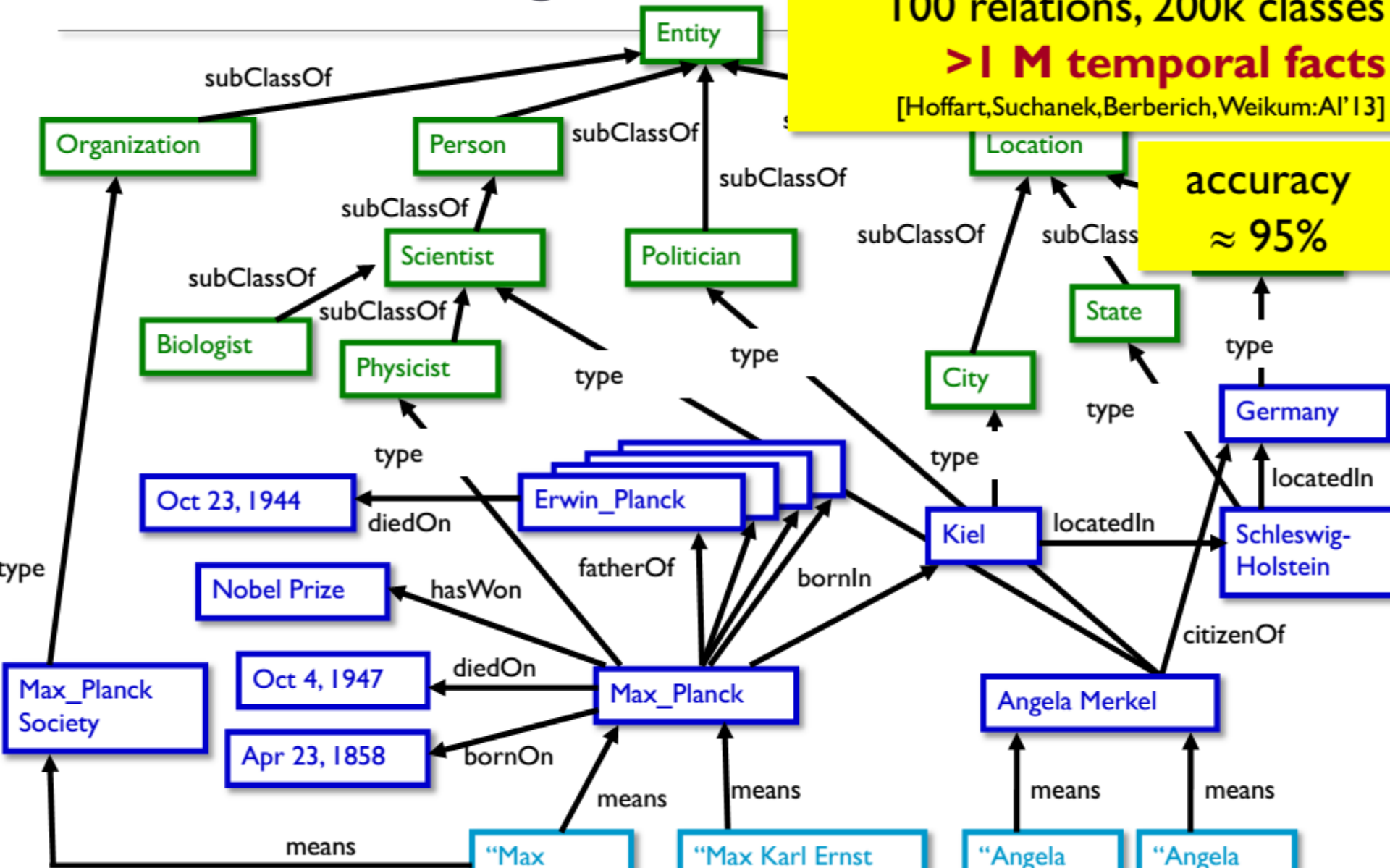
dbo:anthem	dbr:The_Star-Spangled_Banner
dbo:areaTotal	9833516638013.326172 (xsd:double) 9833517000000.000000 (xsd:double)
dbo:capital	dbr:Washington,_D.C.
dbo:demonym	American (en)
dbo:ethnicGroup	dbr:White_American dbr:Native_American dbr:African-American dbr:Asian_American dbr:Multiracial_American
dbo:flag	Flag of the United States.svg

dbp:areaLabel	Total area Total land area
dbp:areaMagnitude	1 (xsd:integer)
dbp:areaRank	3 (xsd:integer)
dbp:callingCode	dbr:North_American_Numbering_Plan
dbp:caption	--09-11 One World Trade Center, built in its place
dbp:cctld	dbr:.mil dbr:.edu dbr:.gov dbr:.us

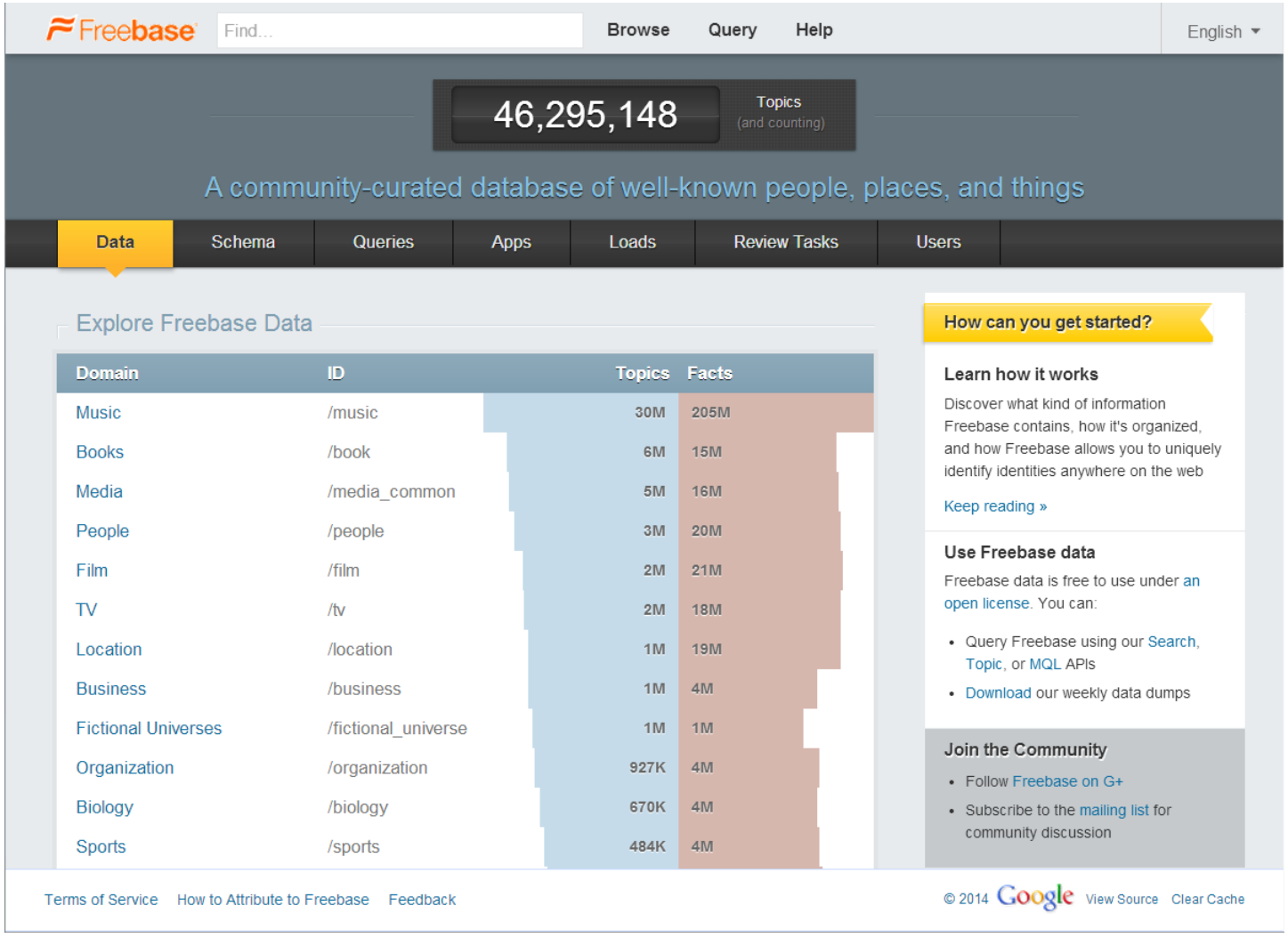
YAGO2 Knowledge Base

3 M entities, 220 M core facts
100 relations, 200k classes
>1 M temporal facts
[Hoffart,Suchanek,Berberich,Weikum:AI'13]

accuracy
≈ 95%

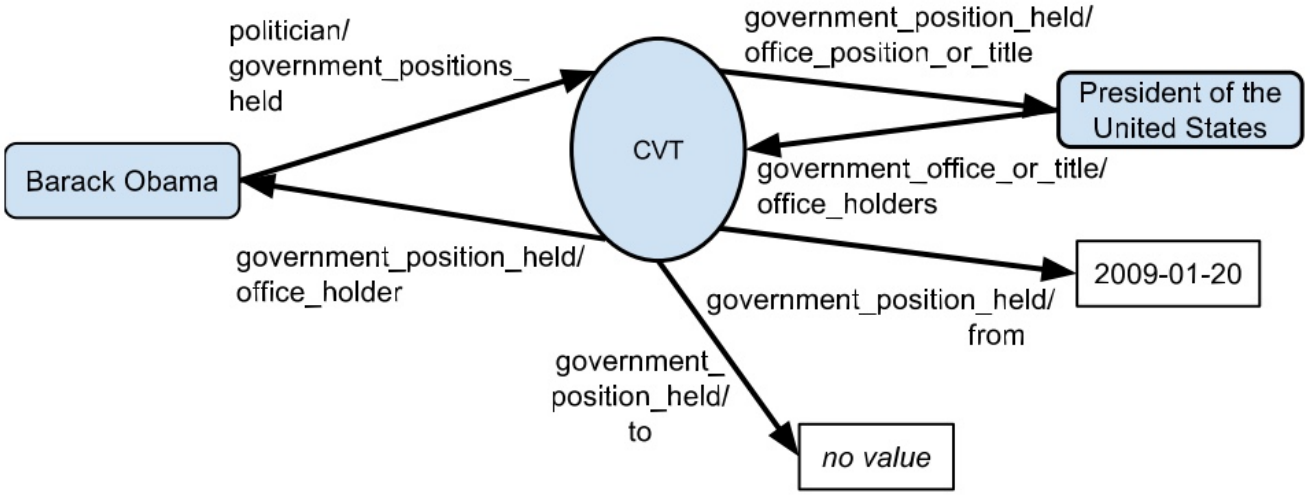


Freebase

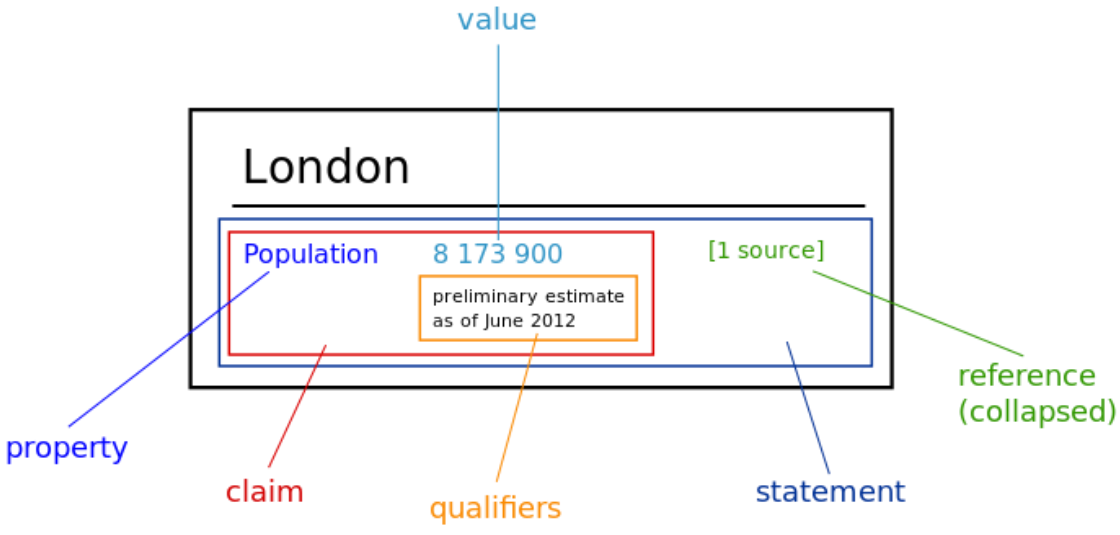

Freebase
Find...
Browse
Query
Help
English
46,295,148
Topics
(and counting)
A community-curated database of well-known people, places, and things
Data
Schema
Queries
Apps
Loads
Review Tasks
Users
Explore Freebase Data
Domain
ID
Topics
Facts
Music /music 30M 205M
Books /book 6M 15M
Media /media_common 5M 16M
People /people 3M 20M
Film /film 2M 21M
TV /tv 2M 18M
Location /location 1M 19M
Business /business 1M 4M
Fictional Universes /fictional_universe 1M 1M
Organization /organization 927K 4M
Biology /biology 670K 4M
Sports /sports 484K 4M
How can you get started?
Learn how it works
Discover what kind of information Freebase contains, how it's organized, and how Freebase allows you to uniquely identify identities anywhere on the web
Keep reading »
Use Freebase data
Freebase data is free to use under an open license. You can:
Query Freebase using our Search, Topic, or MQL APIs
Download our weekly data dumps
Join the Community
Follow Freebase on G+
Subscribe to the mailing list for community discussion
Terms of Service
How to Attribute to Freebase
Feedback
© 2014 Google
View Source
Clear Cache

多元关系-Freebase

- Freebase使用复合值类型 (CVT：Compound Value Types)来处理多元关系。例如下面这个例子中的CVT描述了关于Obama的任职期限的多元关系“government_position_held”。这个多元关系包含多个子二元关系：“office_holder”，“office_position”，“from”，“to”等。一个CVT就是一个有唯一MID的Object，也可以有多个Types。为了以示区别，Freebase把所有非CVT的Object也称为“Topic”。



Wikidata知识表述元素及示例



名称	作用
Entity	顶层对象，类似于OWL：Thing，包含Item和Property
Item	实例对象，类似于instance
Property	属性对象，类似于RDF中的Property
Statement	陈述集合，含有一个属性、多个值对象以及修符饰符和引用符
Qualifier	修饰一个陈述，处理多元复杂关系
Snak	陈述的基本描述结构，类似于RDF三元组
Reference	标识陈述的来源

多元关系-Wikidata

- Wikidata使用修饰-Qualifiers用于处理复杂的多元表示。如一个陈述“spouse: Ivana Trump”描述了一个二元关系。我们可以使用Qualifiers给这个陈述增加多个附加信息来刻画多元关系，如：“start date: 7 April 1977” and “end date: 22 March 1992,”等。
- 引用-References用于标识每个陈述的来源或出处，如来源于某个维基百科页面或特定站点等。引用也是一种Qualifiers，通常添加到Statements的附加信息中。

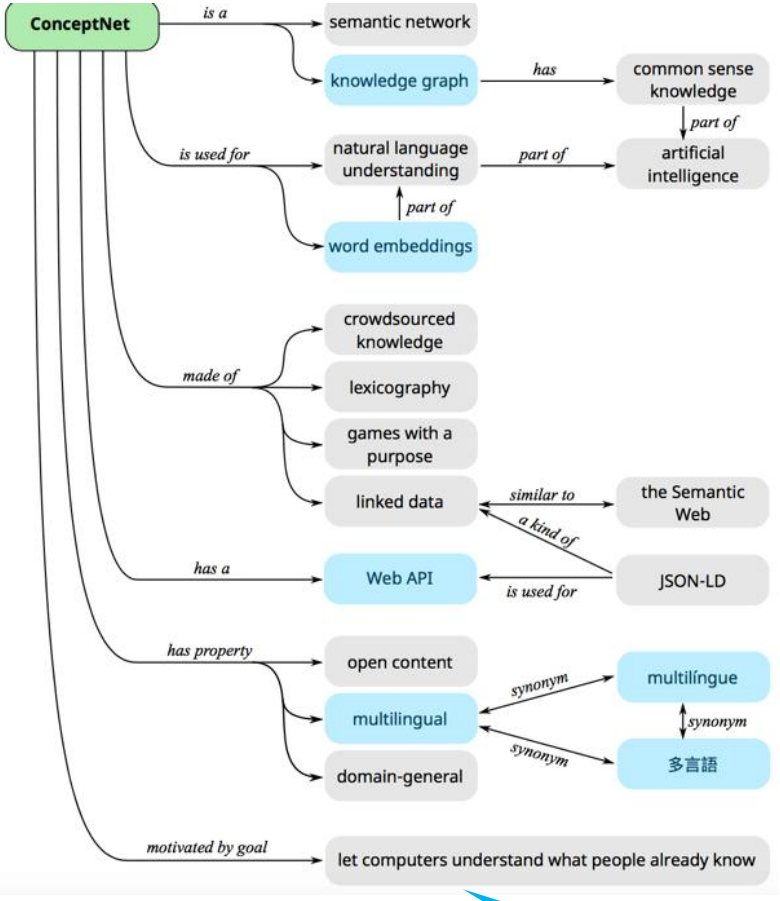
ConceptNet基本表达要素

ConceptNet
An open, multilingual knowledge graph

主要特点：允许Node是自然语言描述，甚至是一个句子，但关系类型相对确定。有利于降低知识获取的难度，更加适合于自然语言处理相关的任务，如文本理解，问答等。

Relation	Sentence pattern	Relation	Sentence pattern
IsA	*NP* is a kind of *NP*.	LocatedNear	You are likely to find *NP* near *NP*.
UsedFor	*NP* is used for *VP*.	DefinedAs	*NP* is defined as *NP*.
HasA	*NP* has *NP*.	SymbolOf	*NP* represents *NP*.
CapableOf	*NP* can *VP*.	ReceivesAction	*NP* can be *VP*.
Desires	*NP* wants to *VP*.	HasPrerequisite	*NP*|*VP* requires *NP*|*VP*.
CreatedBy	You make *NP* by *VP*.	MotivatedByGoal	You would *VP* because you want *VP*.
PartOf	*NP* is part of *NP*.	CausesDesire	*NP* would make you want to *VP*.
Causes	The effect of *VP* is *NP*|*VP*.	MadeOf	*NP* is made of *NP*.
HasFirstSubevent	The first thing you do when you *VP* is *NP*|*VP*.	HasSubevent	One of the things you do when you *NP*|*VP*.
AtLocation	Somewhere *NP* can be is *NP*.	HasLastSubevent	The last thing you do when you *VP* is *I*
HasProperty	*NP* is *AP*.		

Table 1: The interlingual relations in ConceptNet, with example sentence frames in English.

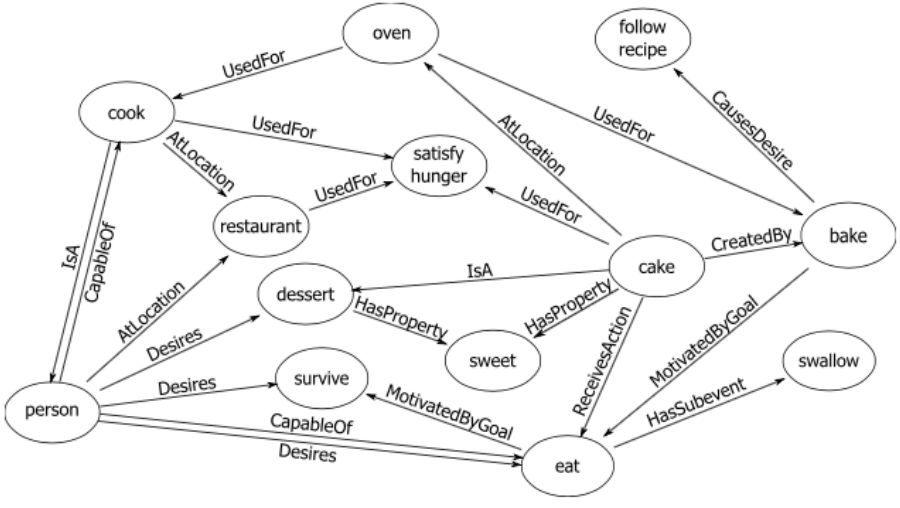


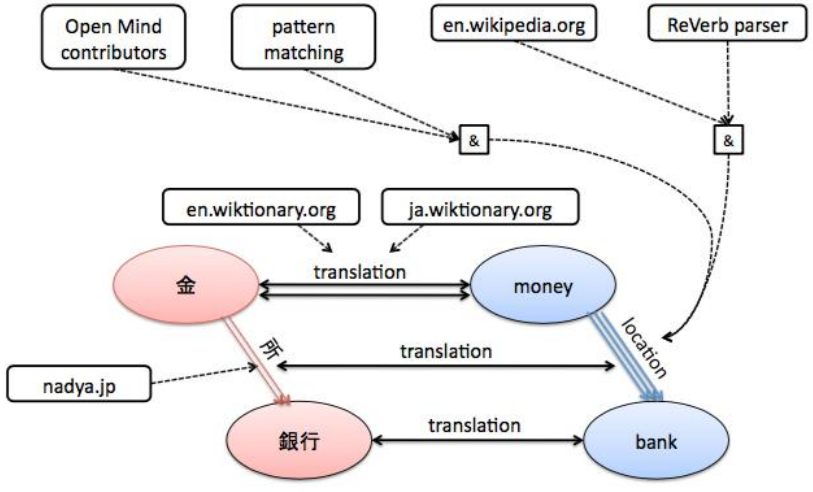
NODE可以是自由文本

概念-Concepts、词-Words、短语-Phrases、断言-Assertions、关系-Relation、边-Edges

多元关系-ConceptNet5

- ConceptNet5的知识表示框架主要包含如下要素：概念-Concepts、词-Words、短语-Phrases、断言-Assertions、关系-Relations、边-Edges。
- Assertions描述了Concepts之间的关系，类似于RDF中的Statements。Edges类似于RDF中的Property。一个Concepts包含多条边，而一条边可能有多个产生来源。例如，一个“化妆 Cause 漂亮”的断言可能来源于文本抽取，也可能来源于用户的手工输入。来源越多，该断言就越可靠。
- 在处理表示“x is the first argument of y”这类多元关系的问题上，ConceptNet5把所有关于某条边的附加信息增加为边的属性。





NELL抽取结果示例

Read the Web

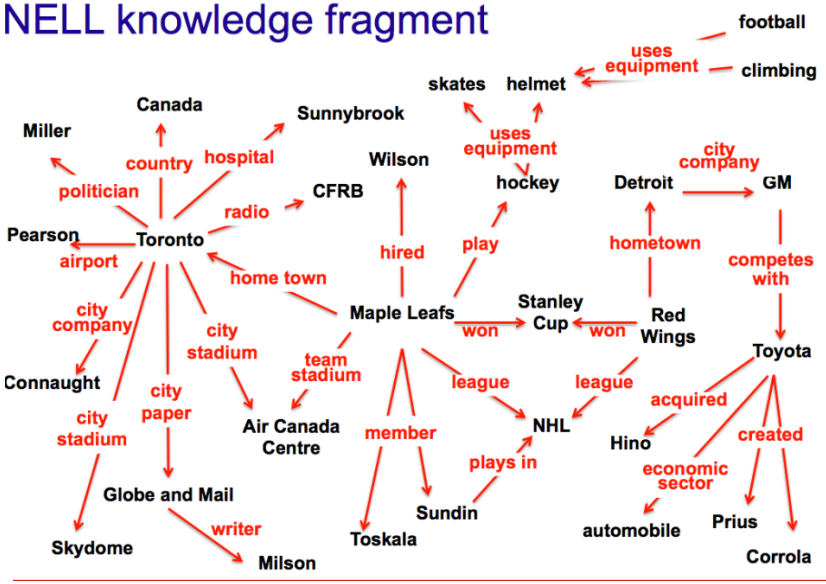
Recently-Learned Facts

twitter

Refresh

instance	iteration	date learned	confidence
n1_3_problems is a song	1075	04-sep-2017	98.5
research_programmes is a research project	1075	04-sep-2017	100.0
monetary_damage is an event outcome	1075	04-sep-2017	92.6
metal_coffee_table is a household item	1075	04-sep-2017	100.0
n7_8_minutes is a term used by physicists	1075	04-sep-2017	100.0
groupis part of wheel	1075	04-sep-2017	93.8
ben_gordon is an athlete coached by scott_stiles	1079	28-sep-2017	100.0
beds is often found in story	1078	23-sep-2017	93.8

NELL knowledge fragment



NELL Knowledge Base Browser

Search

log in | preferences | help/instructions | feedback

categories | relations

- everypromotedthing
  - abstractthing
    - creativework
      - book
      - poem
      - lyrics
      - musicalbum
      - musicsong
      - televisionshow
      - movie
      - visualartform
    - celltype
    - physicalaction
    - species
      - animal
        - vertebrate
          - bird
          - fish
          - reptile
          - mammal
          - amphibian
        - invertebrate
          - arthropod
            - insect
            - crustacean
            - arachnid
          - mollusk

book
(category)

A book is a written work or a collection of written works in book form. \Book\ represents the abstract notion of a particular book, rather than a particular edition.

View list | map | metadata
6,750 instances, 2 pages: 1 2 next

instance	iteration	date learned	confidence
about_a_boy	599	22-jun-2012	100.0
across_the_river_and_into_the_trees	839	22-may-2014	100.0
airframe	551	19-apr-2012	100.0
airman	649	21-oct-2012	100.0
alice_in_wonderland	602	27-jun-2012	100.0
alice_s_adventures_in_wonderland	670	17-dec-2012	100.0
all_around_the_world	839	22-may-2014	100.0
all_quiet_on_the_western_front	670	17-dec-2012	100.0
all_the_king_s_men	222	21-mar-2011	100.0
all_through_the_night	838	19-may-2014	100.0
all_together_dead	551	19-apr-2012	100.0
all_you_need_is_love	844	06-jun-2014	100.0
along_came_a_spider	551	19-apr-2012	100.0
already_dead	551	19-apr-2012	100.0
altered_carbon	533	18-mar-2012	100.0
amelia	670	17-dec-2012	100.0
american_gods	218	13-mar-2011	100.0
american_pastoral	348	14-jul-2011	100.0
american_psycho	599	22-jun-2012	100.0
american_tabloid	533	18-mar-2012	100.0

知识表示：多元关系

名称	多元关系表述
DBPedia	无考虑，可通过Blank Node等用多个三元组来表示
Freebase	CVT复合类型节点
WikiData	Qualifier或者Reference
ConceptNet	将多元关系添加为边的属性

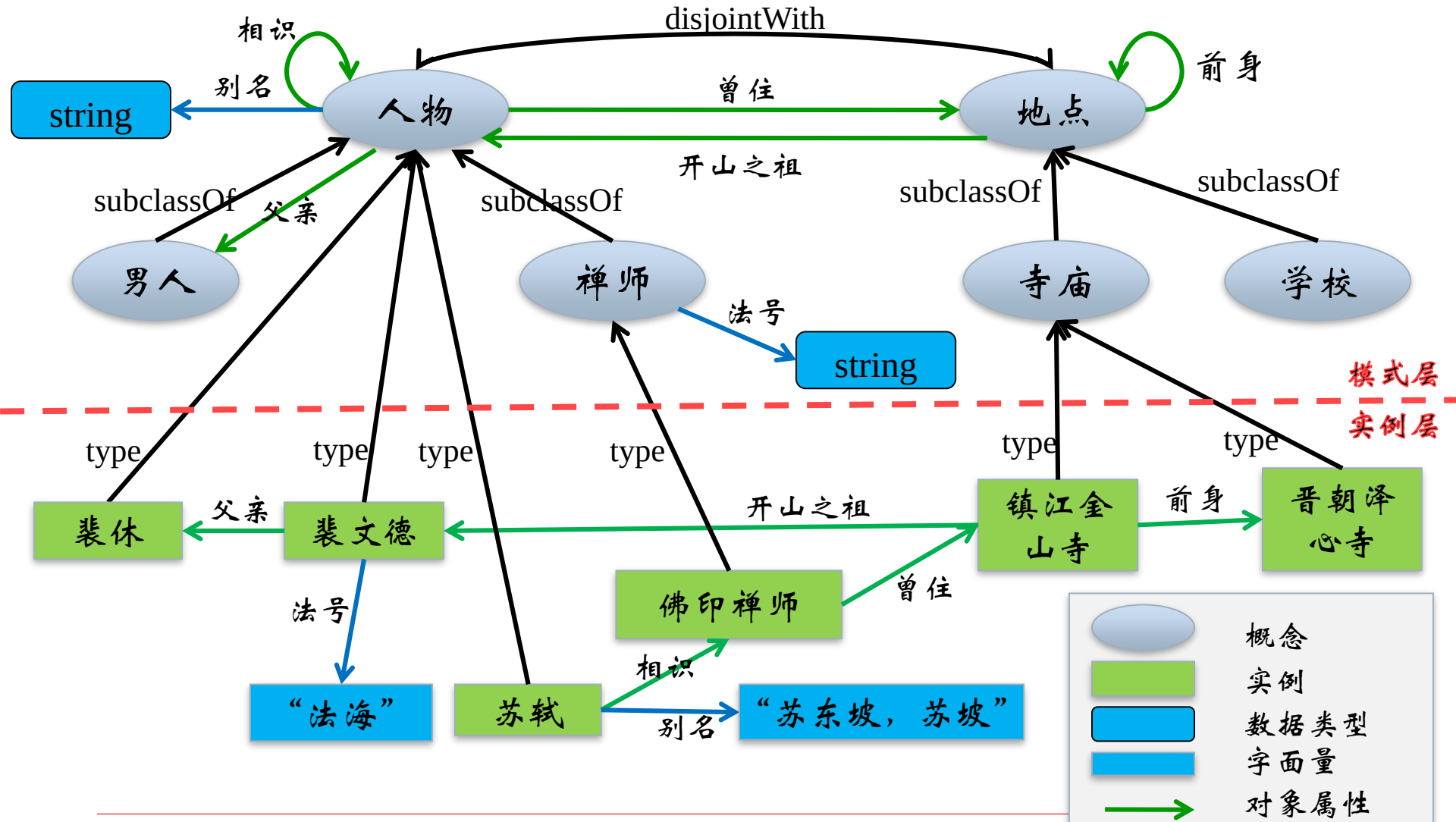
实用的知识表示

- 接近人的自然语言：好的KR是同时为机器和人设计的
- 够用的表达能力：够用就好，不必苛求逻辑完备
- 易于扩展：能够非常方便的增加新的类、实体和关系

RDF/OWL只是众多知识表示框架之一，不少商业化的知识图谱并未采用RDF/OWL，物理存储也直接采用关系数据库实现，但基本表达要素都可以在RDF/OWL中找到对应。

第四部分：基于 Protégé的知识建模实践

Protégé使用——知识图谱示例



Protégé使用

Protégé简介

- Protégé软件是斯坦福大学医学院生物信息研究中心基于Java语言开发的本体编辑和本体开发工具，也是基于知识的编辑器，属于开放源代码软件

- 这个软件主要用于语义网中本体的构建，是语义网中本体构建的核心开发工具，现在的最新版本为5.2.0版本[1]（截止2017年9月29日）

[1] https://protege.stanford.edu/products.php

Protégé使用

Protégé特点

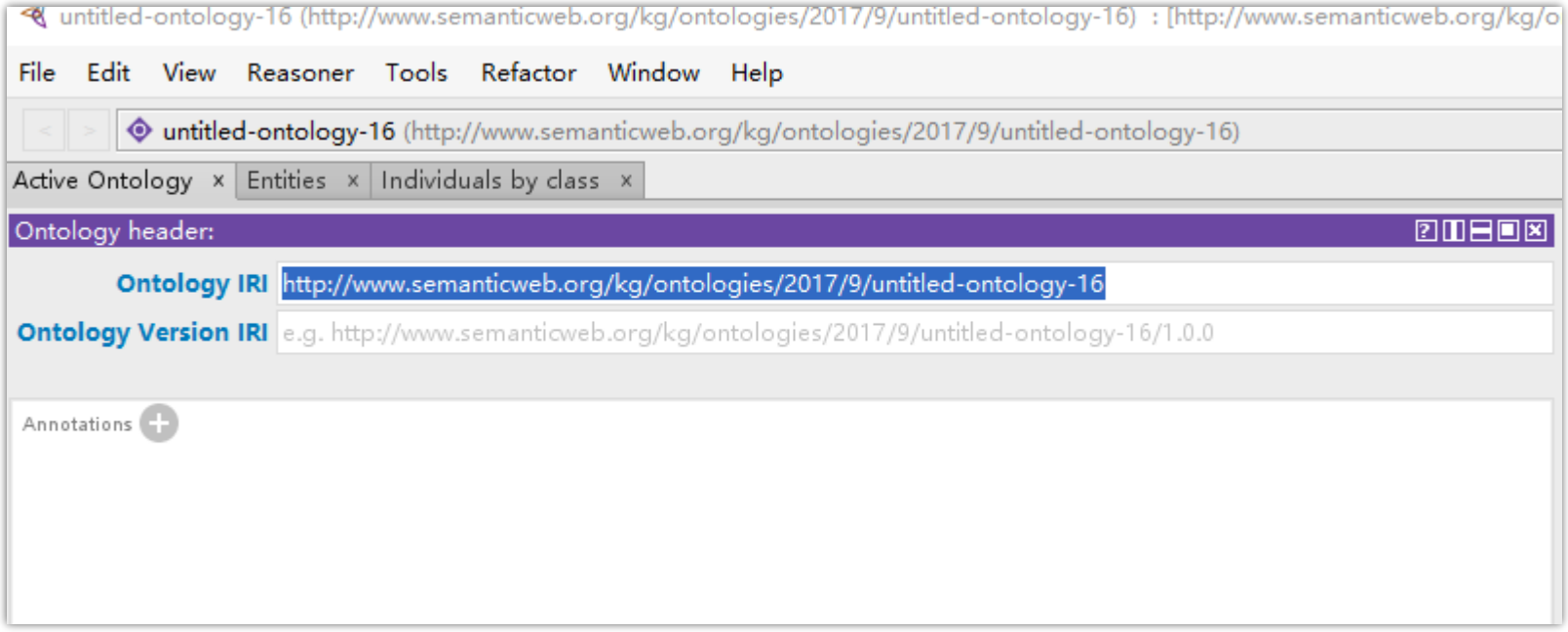
- Protégé是一组自由开源的工具软件，用于构建域模型与基于知识的本体化应用程序。
- Protégé提供了大量的知识模型架构与动作，用于创建、可视化、操纵各种表现形式的本体。
- 可以通过用户定制实现域-友好(领域相关)的支持，用于创建知识模型并填充数据。
- Protégé可以通过两种方式进行扩展：插件和基于java的API。
- 相比与其他的本体构建工具而言，Protégé最大的好处在于支持中文，在插件上，用OntoGraf可实现中文关系的显示。

Protégé使用

Protégé用途

- 类建模 (Class modeling)： Protégé提供了一个图形化用户界面来建模类（领域概念）和它们的属性及关系。
- 实例编辑 (Instance editing)：从这些类中， Protégé自动产生交互式的形式，全用户或领域专家进行有效实例编辑成为可能。
- 模型处理 (Model processing)：Protégé有一些插件库，可以定义语义、解答询问以及定义逻辑行为。
- 模型交换 (Model exchange)：最终的模型 (类和实例)能以各种各样的格式被装载和保存，包括XML、UML和资源描述框架RDF。

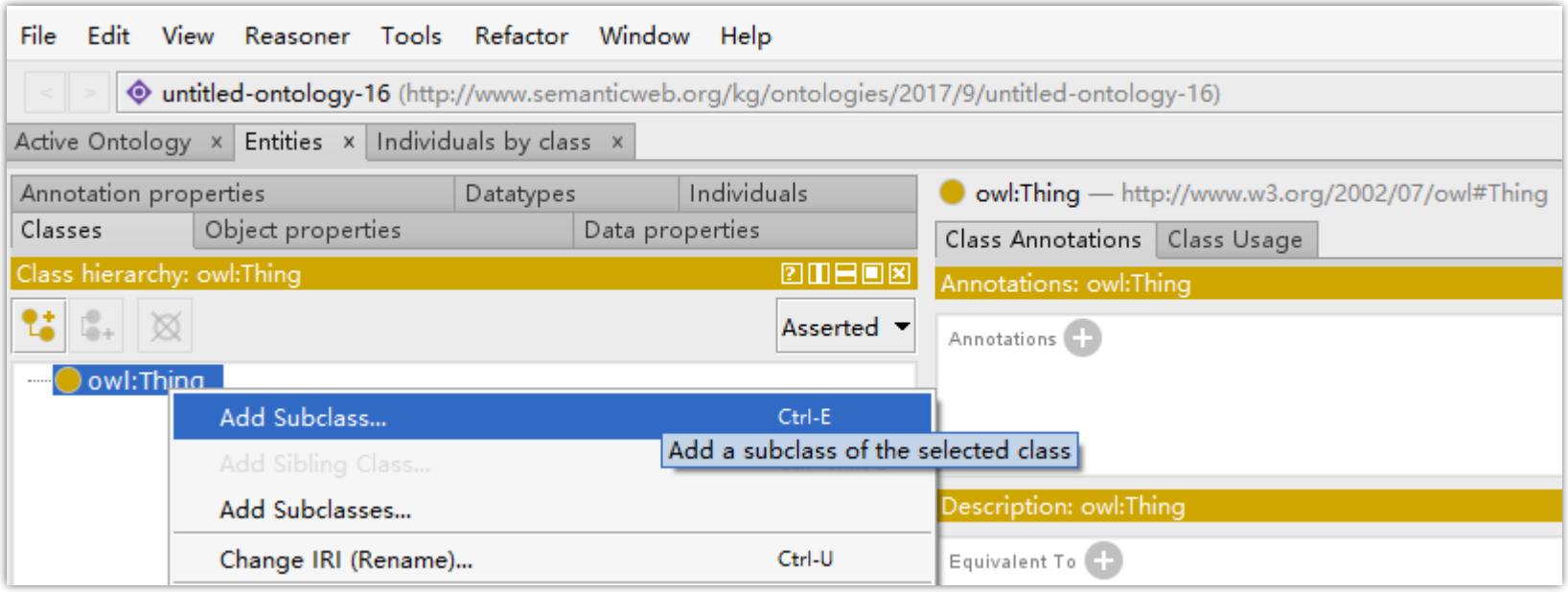
Protégé使用



步骤1 建立新的本体。

- 打开Protégé软件后 (界面显示如上图)，便是新建本体的界面；
- 或者，可以在菜单里面选择File→ New… 新建一个本体。

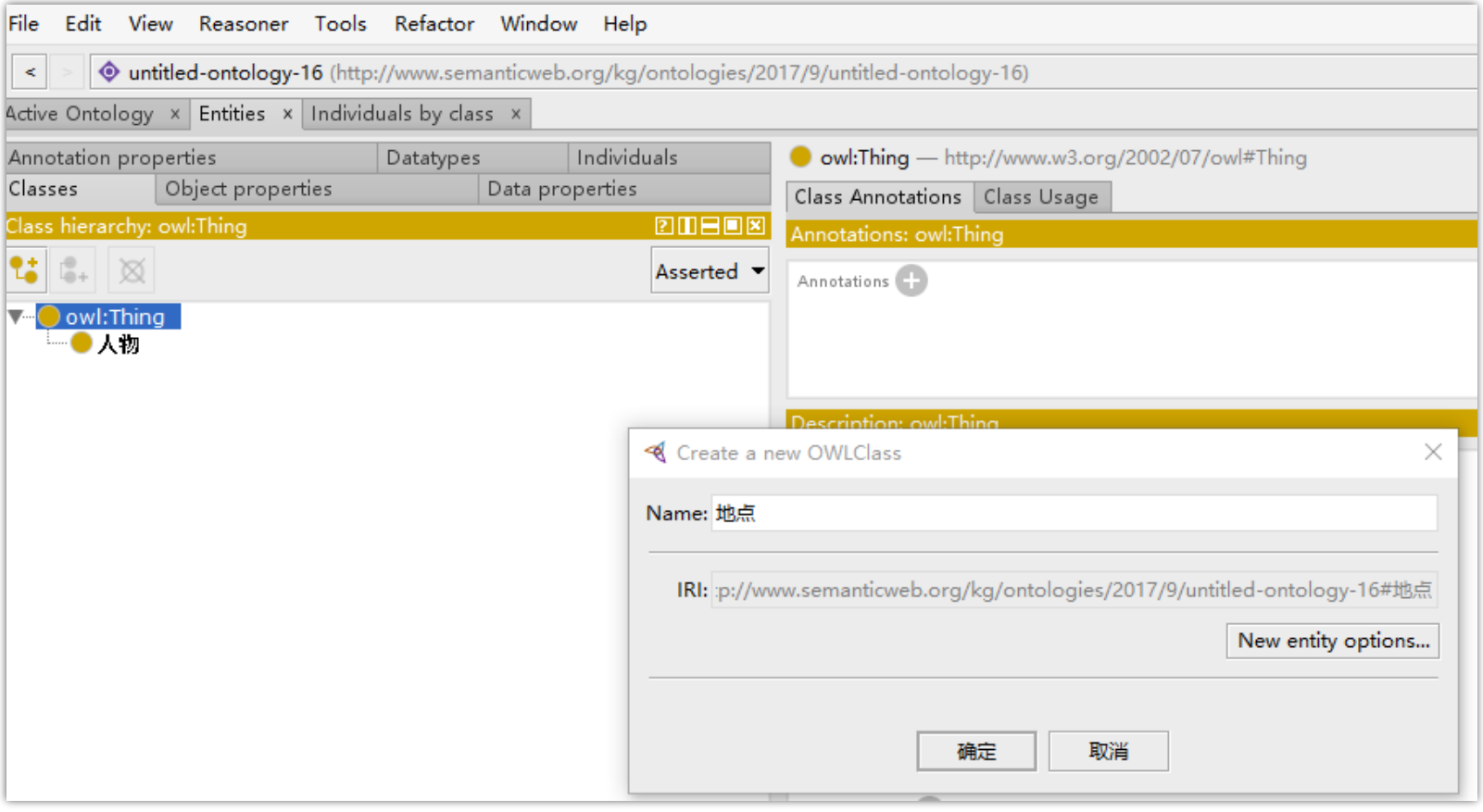
Protégé使用



步骤2 构建类。

- Protégé的主页面中，点击Entities页面，进入本体的编辑界面；
- 在Entities页面，选择Classes标签（默认情况下用户看到的是该标签的页面），进入类及其层次的编辑页面；
- 在Classes页面，右键点击owl:Thing，选择Add Subclasses…，在出现的对话框中Name标签后输入类的名字，然后点击确定；
- 在Classes页面点击一个类名，在Protégé右侧Description部分修改跟该类相关的属性值

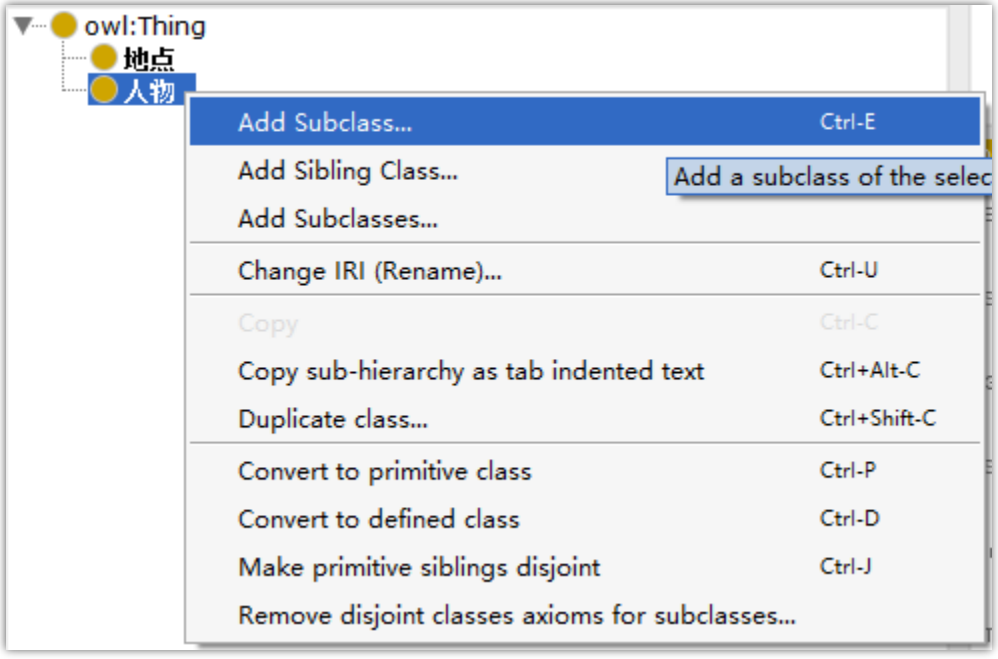
Protégé使用



步骤2 构建类。

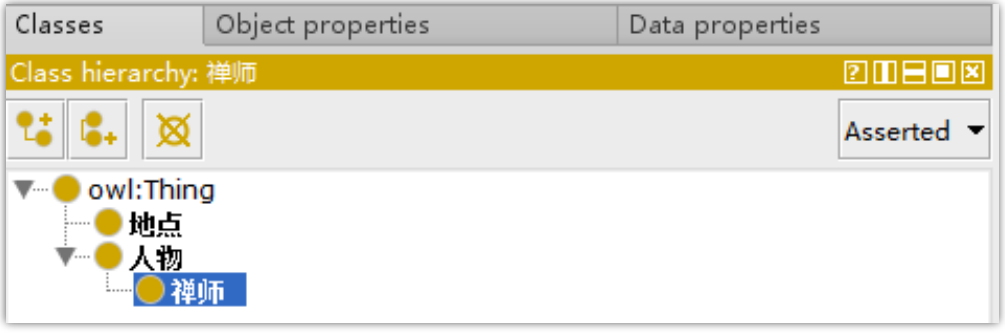
例如：构建类“地点”

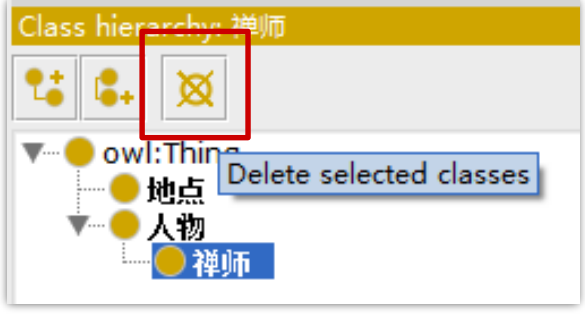
Protégé使用



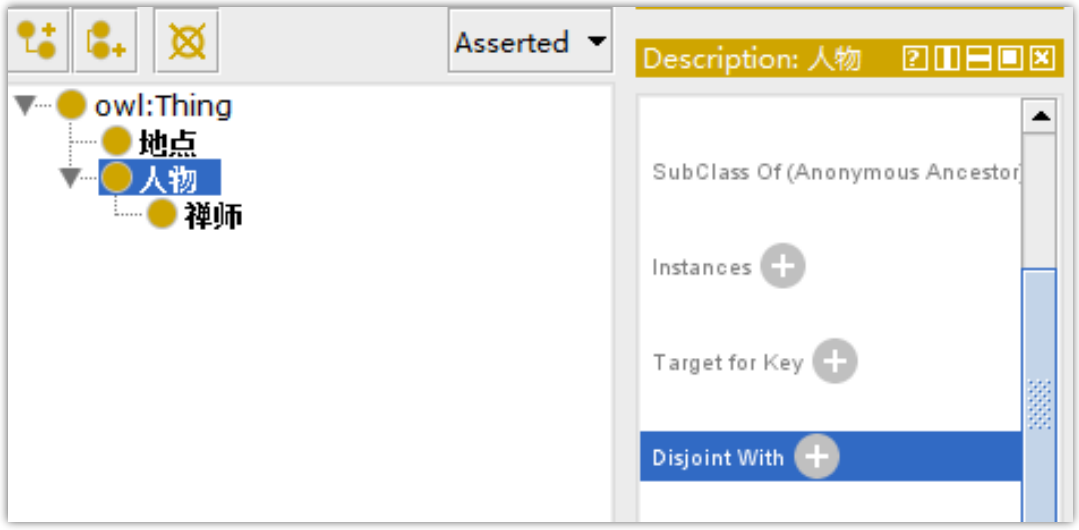
步骤3 建立子类。

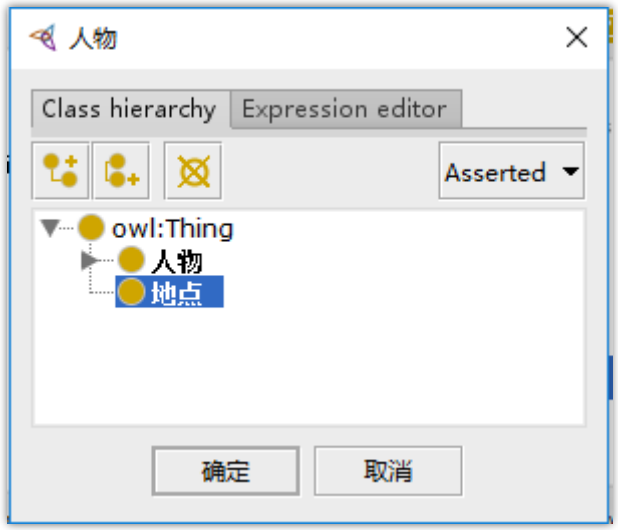
- 在“人物”上右键点击，选择Add subclass…；
- 在弹出的对话框中输入子类名称，如“禅师”，点击确定，在Classes界面显示类的层次(见左下角图)；
- 如果需要删除某个类，点击该类，然后点击下图红框里面图案。





Protégé使用





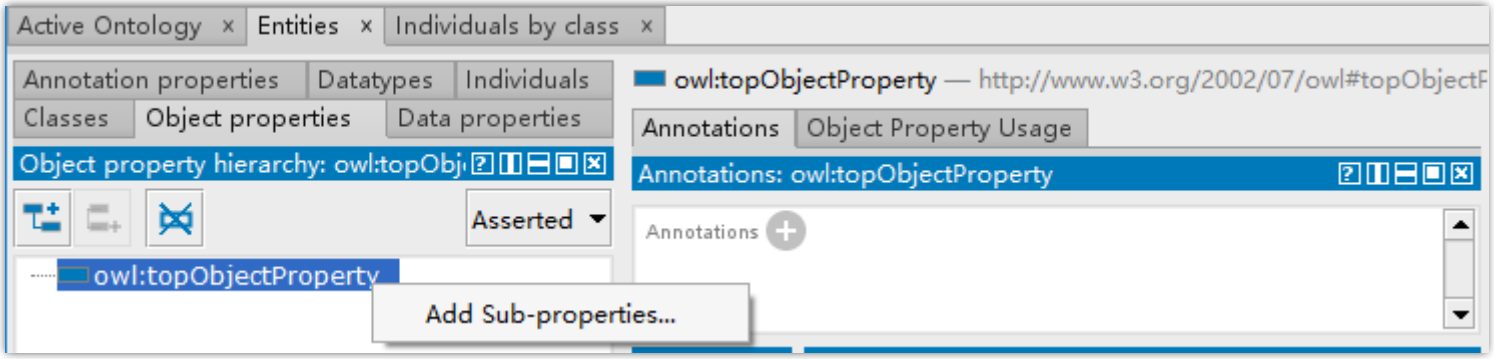
步骤4 构建类之间的关系。

因为人物和地点是不同的事物，即它们互相具有排他性(owl:disjointWith)，下面定义该关系。

- 在选中“人物”的状态下，在Entities界面右侧Description部分点击DisjointWith后的加号（见上面左图）；
- 在弹出的界面中（见上面右图），展开owl:Thing，选择“地点”，然后确定。

这样人物和地点就有互相排斥的属性了。

Protégé使用

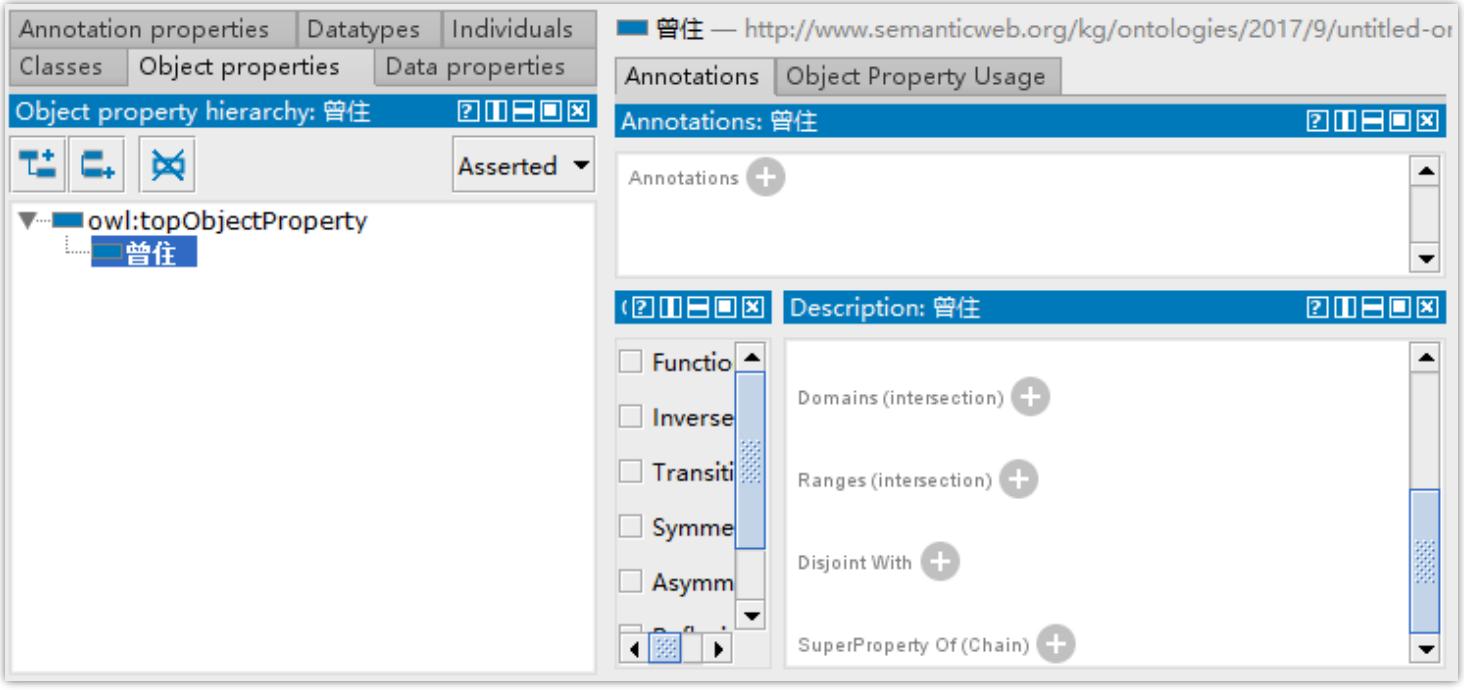


步骤5 建立对象属性。

添加属性名

- 在Entities界面选择Object properties标签，进入对象属性的编辑界面(上图)；
- 在owl:topObjectProperty上右键点击，选择”Add Sub-properties…”；
- 在弹出的界面中(左图)，输入属性名称，例如“曾住”，点击“Continue”，再点“Finish”，便建好属性；

Protégé使用

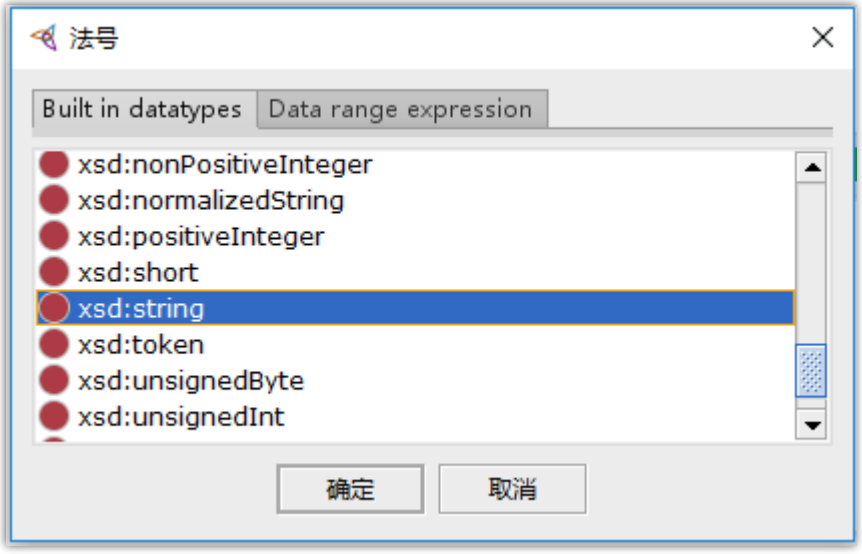


步骤5 建立对象属性。

为属性添加domain和range属性值

- 在Object properties界面，选择一个属性，例如“曾住”；
- 在Entities界面的右侧Description模块中点击Domain后的加号；
- 在弹出的界面中选择“人物”，点击确定，这样便为“曾住”加了domain的约束；
- 类似地，点击Description中的Range后的加号，选择“地点”。

Protégé使用

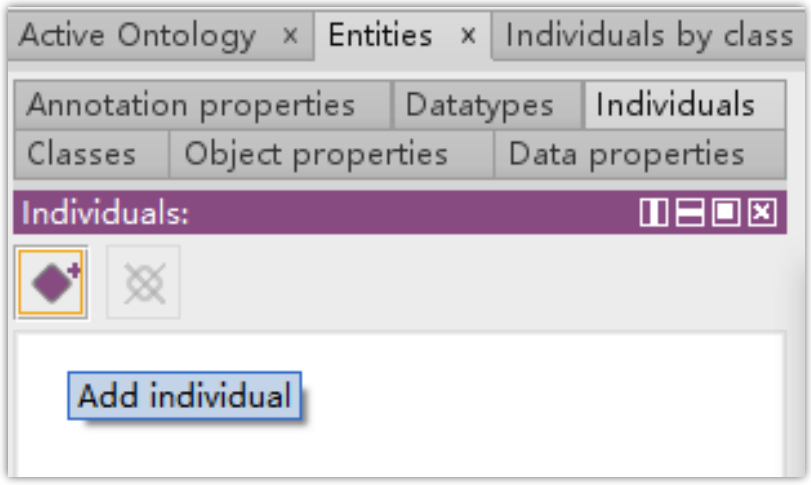


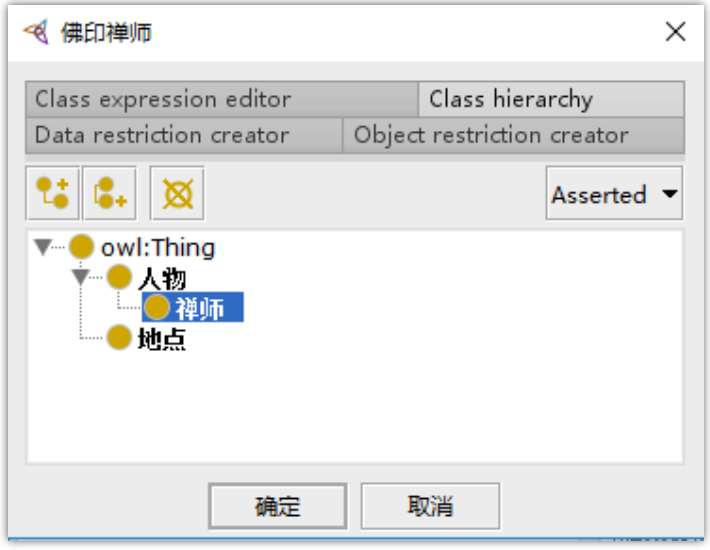
步骤6 建立数据属性。

- 在Entities界面选择Data properties标签，进入数据属性的编辑界面；
- 在owl:topDatatProperty上右键点击，选择”Add Sub-properties…”；
- 在弹出的界面中（左图），输入属性名称，例如“法号”，点击“Continue”，再点“Finish”，便建好属性；
- 在Data properties界面，选择一个属性，例如“法号”；
- 在Entities界面的右侧Description模块中点击Range后的加号；
- 在弹出的界面中，选择“Built in dataypes”，从中挑选xsd:string，再点击确定，即限制该属性的取值范围是字符串。

建立数据属性类似于建立对象属性，主要是在加range约束时不同。

Protégé使用





步骤7 建立实例。

添加实例及其类型

- 在Entities界面选择Individuals标签，进入实例的编辑界面(见上面左图)；
- 单击Individuals界面的菱形图标，在弹出的界面输入实例名字，如“佛印禅师”，点确定。
- 在Entities的右侧界面Description部分，点击Types后面的加号(见左图)，在出现的界面中选择Class Hierarchy标签，从类层次中选中“禅师”。这样该实例就有了类型约束。

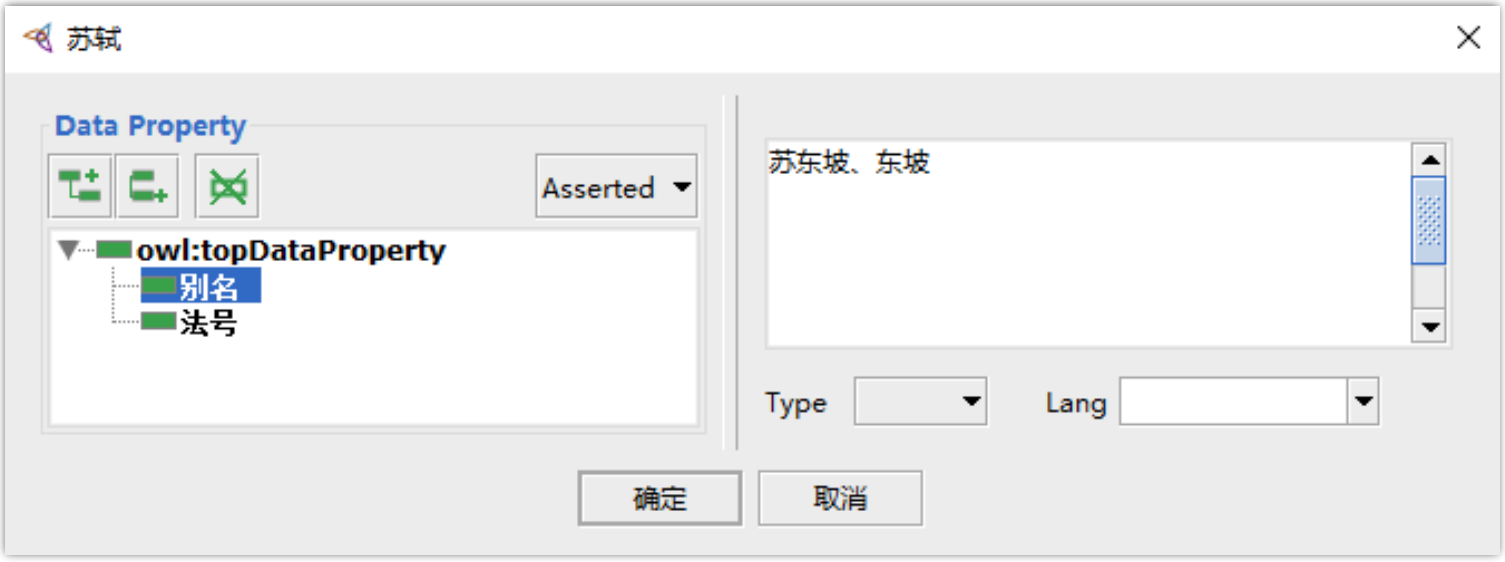
Protégé使用

步骤7 建立实例。

添加实例之间的关系，以“佛印禅师”为例

- 在Individuals界面选择实例“佛印禅师”，在Entities界面右侧的Property assertions部分点击Object property assertions旁的加号(上图)；
- 在弹出的界面分别输入一个对象属性名字(如“曾住”)和一个实例名字(如“镇江金山寺”)，点确定。这样，使得“佛印禅师”与“镇江金山寺”通过“曾住”关联起来了。

Protégé使用

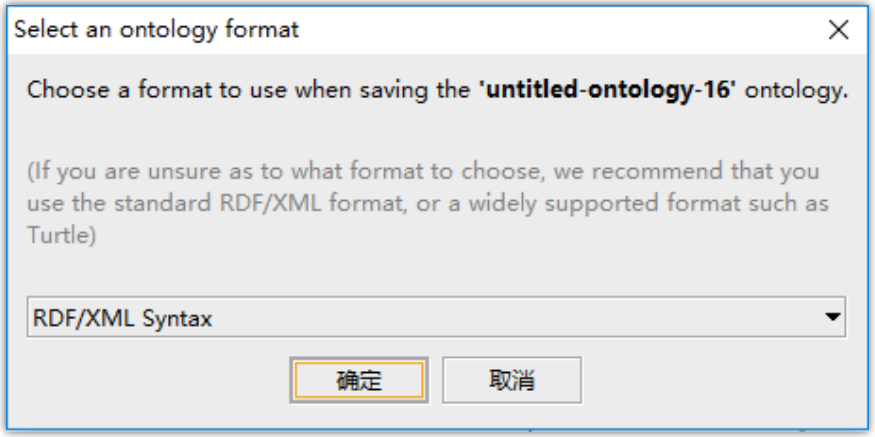


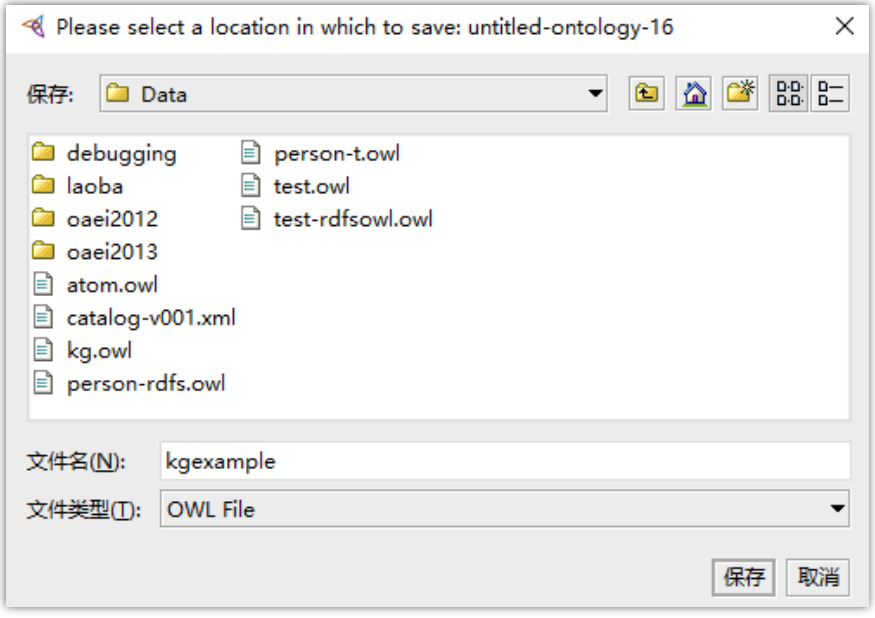
步骤7 建立实例。

添加实例添加属性值，以“苏轼”为例

- 在Individuals界面选择实例“苏轼”，在Entities界面右侧的Property assertions部分点击Data property assertions旁的加号(类似添加实例间关系的操作)；
- 在弹出的界面中(见上图)，在左侧选择数据属性(如“别名”)，右侧填写属性值(如“苏东坡、东坡”)，点确定。这样，便为实例“苏轼”的属性“别名”添加了具体的值，即“苏东坡、东坡”。

Protégé使用

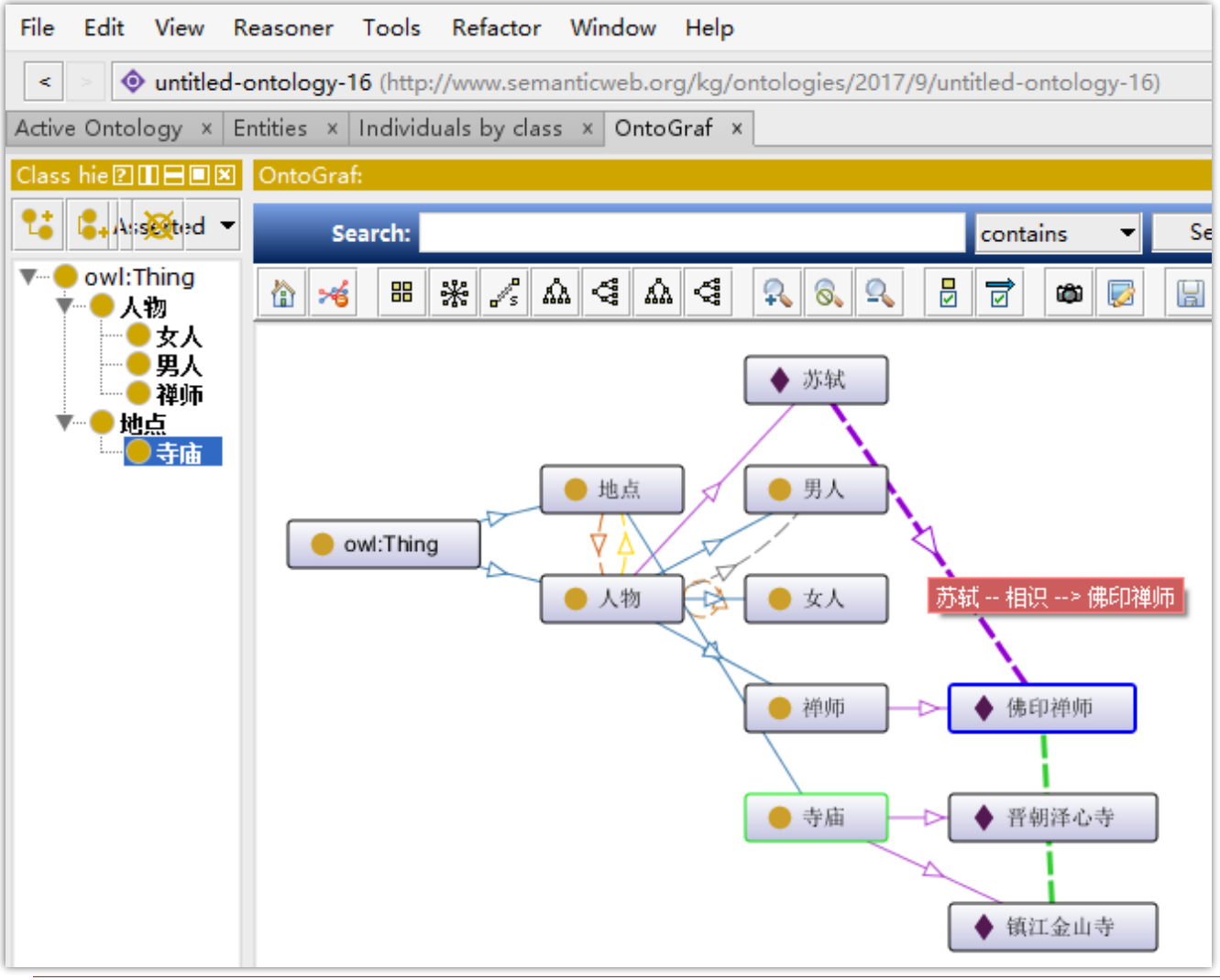




步骤8 保存本体。

- 在菜单选择File→ Save；
- 在弹出的界面中（见左上图），选择“RDF/XML Syntax”，点确定；
- 在又出现的界面中，在文件名处输入本体的名字，例如kgexample，文件类型是“OWL File”，点击保存。

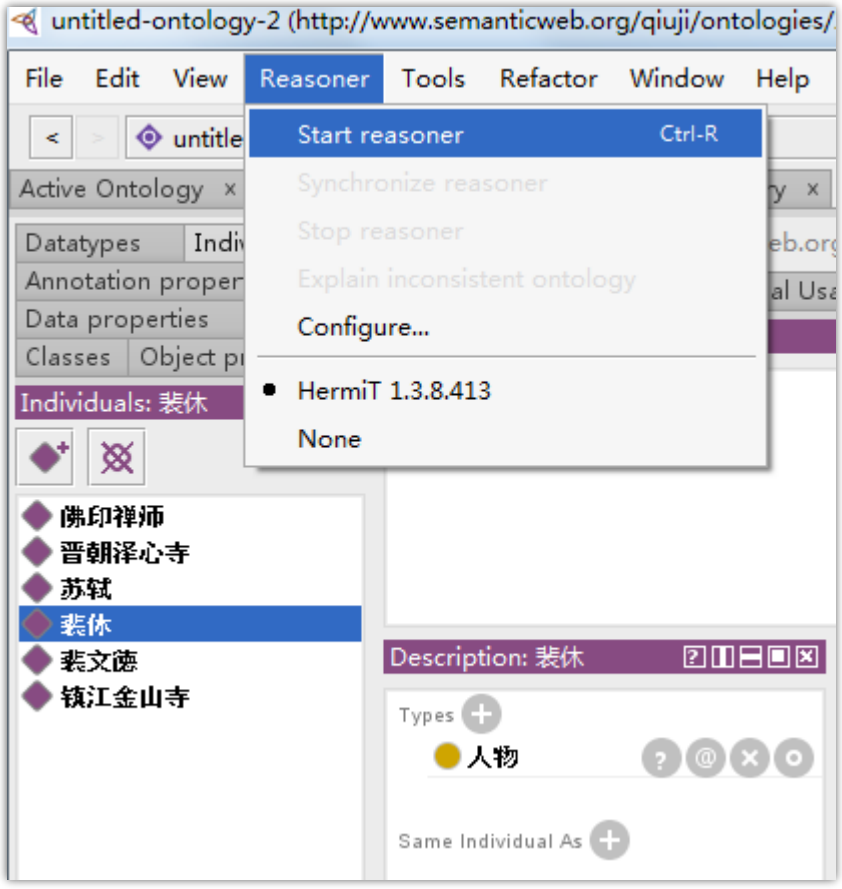
Protégé使用



步骤9 可视化。

- 在菜单中选择Windows–>Tabs→OntoGraf;
- 在出现的界面中，点击加号可以展开，鼠标移到线上，可显示此线代表的关系名称。

Protégé使用

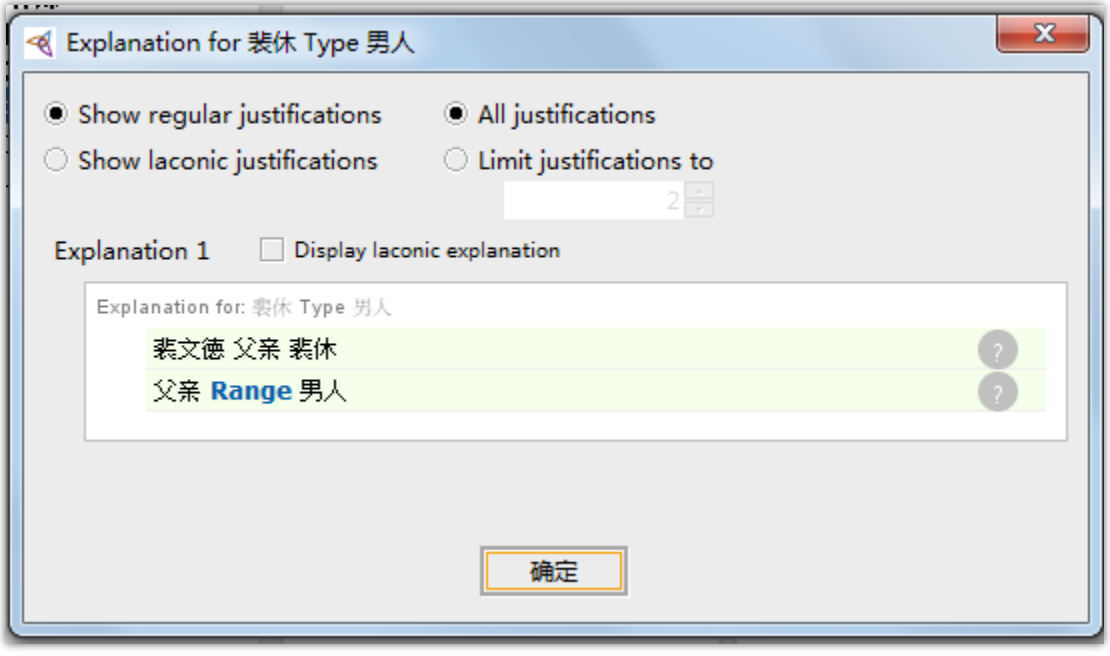


步骤10 推理。

- 在菜单中选择Reasoner
- 在出现的界面中，选择HermiT，然后点击Start reasoner (见左图)。
- 推理得到的信息就会在对应的描述中显示出来。

例如，本体中给出裴文德和裴休是人物的实例，裴休是裴文德的父亲，父亲的domain是人物，range是男人，因此可以推导出裴休是男人的一个实例 (见下图黄色背景部分是推导出来的)。

Protégé使用



步骤10 推理。

- 对于推导出来的信息，如果想知道为什么能被推理机推导出来，可以点击推导出的信息后面的问号(左上图)；
- 解释的原因会在新的对话框中显示(见左下图)。

谢谢大家！

本课程课件由OpenKG提供支持