



语义网与知识图谱

上海大学计算机学院

主讲：刘 炜





RDFS及其形式语义

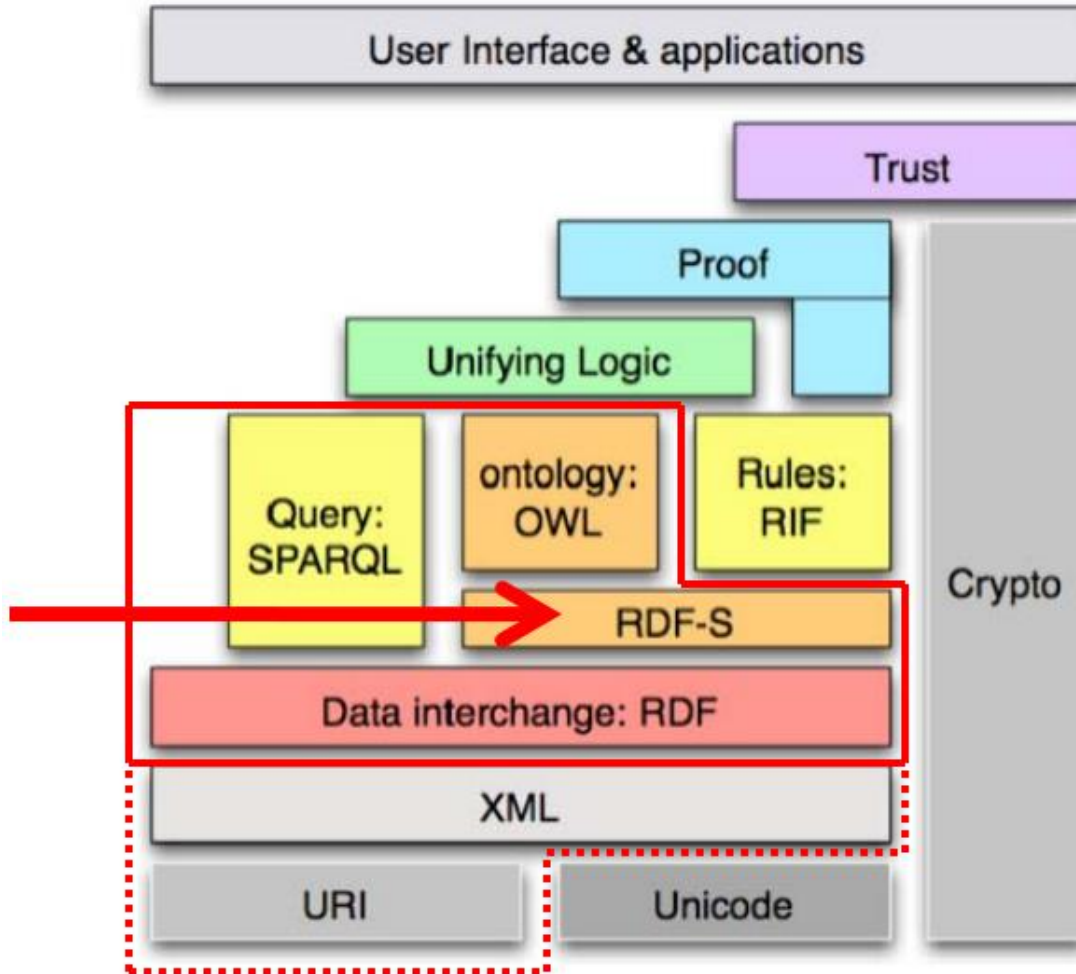
上海大学计算机学院 刘炜

9/14/2020

一、RDFS

二、RDF(S)形式语义

RDF Schema





RDF Schema知识点

- **1. 动机**
- **2. 类和类层次结构 Classes and Class Hierarchies**
- **3. 属性和属性层次结构 Properties and Property Hierarchies**
- **4. 属性约束 Property Restrictions**
- **5. 开放列表回顾 Open Lists Revisited**
- **6. 关于命题的命题：物化 Reification**
- **7. RDFS附加信息 Supplementary Information in RDFS**
- **8. RDFS中的简单本体 Simple Ontologies in RDFS**

- **RDF可用于表示事实 (facts)**
 - Anne is the mother of Merula
- **但是，我们希望能够表示更一般化的知识**
 - Mothers are female
 - If somebody has a daughter then that person is a parent
- **这种知识通常叫做模式(Schema)知识或术语化知识 (terminological knowledge).**
- **RDF Schema允许我们进行模式知识建模，而OWL则具有更好的表示能力。**



RDF Schema概述

- **W3C推荐的资源描述框架部分**
- **用于表示模式/术语化知识**
- **使用RDF提前预定义语义的词汇**
- **每个RDFS文档也是RDF文档**
- **Namespace: <http://www.w3.org/2000/01/rdf-schema#> - 缩写为 rdfs:**
- **词汇是通用的，没有绑定特定应用领域**
 - **允许部分指定用户自定义词汇的语义**
 - **因此，RDF软件可以正确地解释每个RDF Schema中定义的词汇。**

RDF Schema知识点



- 1. 动机
- 2. 类和类层次结构 **Classes and Class Hierarchies**
- 3. 属性和属性层次结构 **Properties and Property Hierarchies**
- 4. 属性约束 **Property Restrictions**
- 5. 开放列表回顾 **Open Lists Revisited**
- 6. 关于命题的命题：物化 **Reification**
- 7. RDFS附加信息 **Supplementary Information in RDFS**
- 8. RDFS中的简单本体 **Simple Ontologies in RDFS**



类和实例

- 类代表事物的集合 (Sets of things)

在RDF: Sets of URIs

rdf:type: instance of a class

- book:uri is a member of the class ex:Textbook

```
book:uri    rdf:type    ex:Textbook .
```

- 一个URI 可以属于多个类

```
book:uri    rdf:type    ex:Textbook .  
book:uri    rdf:type    ex:WorthReading .
```

- 类可以按层次结构进行组织 :

Each textbook is a book

```
ex:Textbook  rdfs:subClassOf  ex:Book .
```

预定义类

- 每个URI表示一个类，属于rdfs:Class的一个成员

```
ex:Textbook    rdf:type    rdfs:Class .
```

- 因此rdfs:Class也是rdfs:Class的成员

```
rdfs:Class    rdf:type    rdfs:Class .
```

- rdfs:Resource (class of all URIs) 资源类
- rdf:Property (class of all properties) 属性类
- rdf:XMLLiteral RDF唯一的预定义类型
- rdfs:Literal (each datatype is a subclass) 所有字面体的类
- rdf:Bag, rdf:Alt, rdf:Seq, rdfs:Container, rdf:List, rdf:nil 列表类
- rdfs:ContainerMembershipProperty (see later) 容器属性类
- rdfs:Datatype (contains all datatypes – a class of classes) 数据类型类
- rdf:Statement (see later) 物化的三元组的类

隐式知识——实例1

- if an RDFS document contains

```
u    rdf:type    ex:Textbook .
```

- and

```
ex:Textbook    rdfs:subClassOf    ex:Book .
```

- Then

```
u    rdf:type    ex:Book .
```

- 这是一个隐式知识的案例，也是逻辑推理的结果，也成为演绎(deduction)或推理(inference)，不需要显式地声明。
- 推理的逻辑结论由形式语义决定。



隐式知识——实例2

- 从

```
ex:Textbook    rdfs:subClassOf    ex:Book .
```

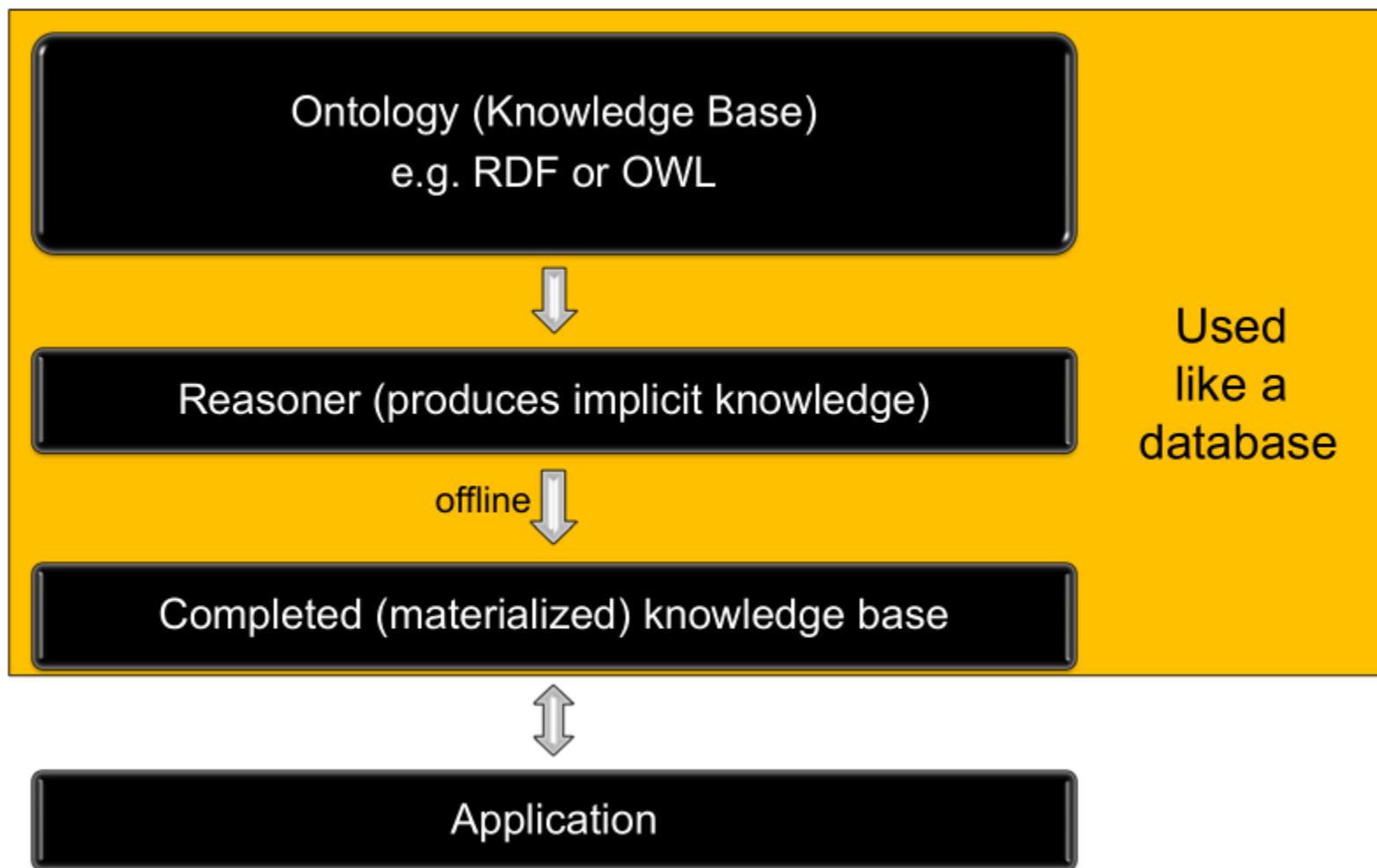
```
ex:Book        rdfs:subClassOf    ex:PrintMedia .
```

- 得出下面的推理结果:

```
ex:Textbook    rdfs:subClassOf    ex:PrintMedia .
```

- **rdfs:subClassOf 是传递的 (transitive) .**

使用隐式知识



类等价



```
ex:MorningStar    rdfs:subClassOf    ex:EveningStar .  
ex:EveningStar    rdfs:subClassOf    ex:MorningStar .
```

类等价：两个类包含相同的个体，互为子类

```
ex:Book    rdfs:subClassOf    ex:Book .
```

Rdfs:subClassOf 关系是自反的，表示每个类是自己的子类。

类和RDF/XML语法



```
<rdf:Description rdf:about= "&ex;SebastianRudolph">  
<rdf:type rdf:resource= "&ex;HomoSapiens">  
</rdf:Description>
```

精简为:

```
<ex:HomoSapiens rdf:about="&ex;SebastianRudolph"/>
```



RDF Schema知识点

- 1. 动机
- 2. 类和类层次结构 Classes and Class Hierarchies
- 3. 属性和属性层次结构 Properties and Property Hierarchies
- 4. 属性约束 Property Restrictions
- 5. 开放列表回顾 Open Lists Revisited
- 6. 关于命题的命题：物化 Reification
- 7. RDFS附加信息 Supplementary Information in RDFS
- 8. RDFS中的简单本体 Simple Ontologies in RDFS

属性层次结构



从

```
ex:isHappilyMarriedTo  rdf:subPropertyOf  ex:isMarriedTo.
```

以及

```
ex:markus    ex:isHappilyMarriedTo    ex:anja .
```

可以推导出:

```
ex:markus    ex:isMarriedTo    ex:anja .
```



RDF Schema知识点

- 1. 动机
- 2. 类和类层次结构 Classes and Class Hierarchies
- 3. 属性和属性层次结构 Properties and Property Hierarchies
- 4. 属性约束 Property Restrictions
- 5. 开放列表回顾 Open Lists Revisited
- 6. 关于命题的命题：物化 Reification
- 7. RDFS附加信息 Supplementary Information in RDFS
- 8. RDFS中的简单本体 Simple Ontologies in RDFS

属性约束

- 可以声明某个属性的类型以及它的取值范围，即定义域和值域。
- 例如： 当 **a is married to b**, 则 **both a and b are Persons**.
- **Expressed by `rdfs:domain` and `rdfs:range`:**

```
ex:isMarriedTo    rdfs:domain    ex:Person .  
ex:isMarriedTo    rdfs:range     ex:Person .
```

- 数据类型限制也是一样:


```
ex:hasAge    rdfs:range    xsd:nonNegativeInteger .
```

陷阱1



定义域和值域之间组成了类和属性之间的语义链接，因为它们提供了描述术语化的相互依赖关系（位于本体成分之间）的唯一方式。但是也会出现一些潜在的错乱。例如：

```
ex:authorOf    rdfs:range    ex:Textbook .  
ex:authorOf    rdfs:range    ex:Storybook .
```

表示作者关系的取值范围同时是课本和故事书（**both**）。

每个被声明的属性限制会全局地影响此属性所有的出现情况，在使用属性限制时要非常小心，尽量使用足够一般的类。

陷阱2



防止出现语义混淆，例如：

```
ex:isMarriedTo    rdfs:domain    ex:Person .
ex:isMarriedTo    rdfs:range     ex:Person .
ex:instituteAIFB  rdf:type       ex:Institution .
```

```
ex:pascal          ex:isMarriedTo ex:instituteAIFB .
```

推理结果如下：

```
ex:instituteAIFB  rdf:type       ex:Person .
```

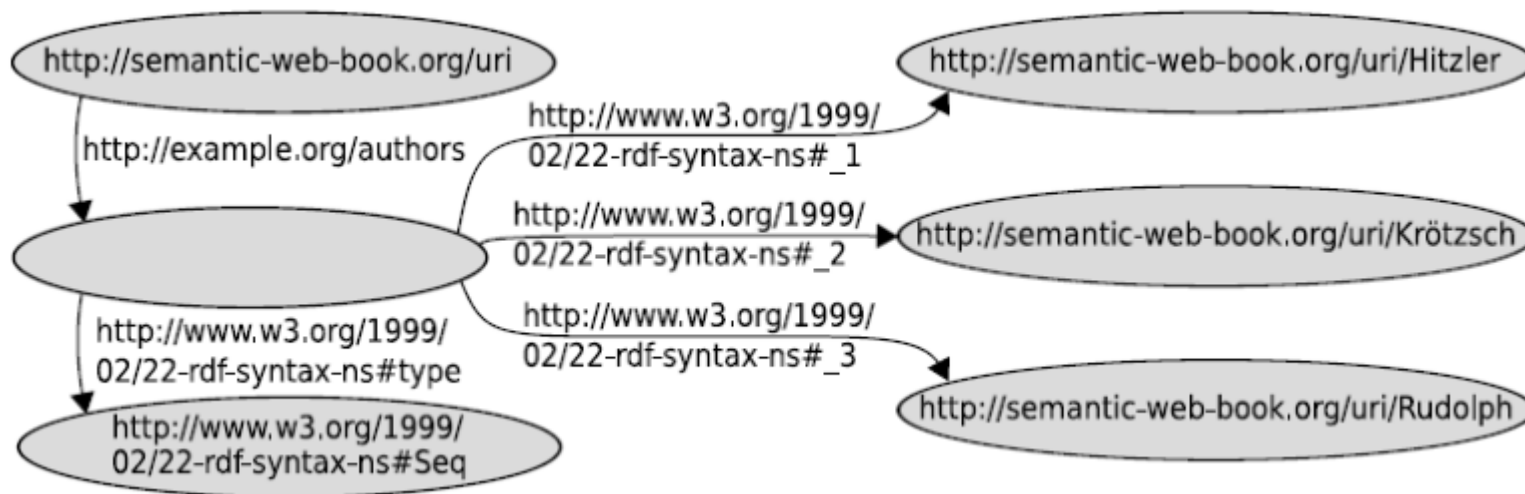
潜在的类
型不匹配
问题，
RDF无法
判定并拒
绝接受



RDF Schema知识点

- 1. 动机
- 2. 类和类层次结构 Classes and Class Hierarchies
- 3. 属性和属性层次结构 Properties and Property Hierarchies
- 4. 属性约束 Property Restrictions
- 5. 开放列表回顾 Open Lists Revisited
- 6. 关于命题的命题：物化 Reification
- 7. RDFS附加信息 Supplementary Information in RDFS
- 8. RDFS中的简单本体 Simple Ontologies in RDFS

开放列表回顾



- 新的类: **rdfs:Container** 是 **rdf:Seq**, **rdf:Bag**, **rdf:Alt**等列表类的超类.
- 新类: **rdfs:ContainerMembershipProperty** 包含用于各种列表容器的属性类, e.g.

```
rdf:_1  rdf:type  rdfs:ContainerMembershipProperty .  
rdf:_2  rdf:type  rdfs:ContainerMembershipProperty .
```

开放列表回顾

- 新属性 `rdfs:member` 是所有包含在 `rdfs:ContainerMembershipProperty` 中的属性的超属性 `superproperty`.
- The RDFS semantics specifies:

from

```
p rdf:type rdfs:ContainerMembershipProperty .
```

and

```
a p b .
```

The following is inferred

```
a rdfs:member b .
```

如果 `rdfs:叔叔` `rdf:type` `rdfs:长辈`
如果 `a` `rdfs:叔叔` `b`
可推出: `a` `rdfs:长辈` `b`



RDF Schema知识点

- 1. 动机
- 2. 类和类层次结构 Classes and Class Hierarchies
- 3. 属性和属性层次结构 Properties and Property Hierarchies
- 4. 属性约束 Property Restrictions
- 5. 开放列表回顾 Open Lists Revisited
- 6. 关于命题的命题：物化 Reification
- 7. RDFS附加信息 Supplementary Information in RDFS
- 8. RDFS中的简单本体 Simple Ontologies in RDFS



回顾一下三元组

- 在一个命题中指向其他命题，非常常见。
- 比如：

“The detective supposes that the butler killed the gardener.”

“侦探推测管家杀死了园丁”，简单建模如下：

- **unsatisfactory:**

```
ex:detective    ex:supposes    "The butler killed the gardener." .
```

通过字面体表达不能被其他三元组引用，因此使用一个**URI**来指定更合理。

```
ex:detective    ex:supposes    ex:theButlerKilledTheGardener .
```

为了避免缺乏结构透明性，将**URI**部分建模成一个独立的三元组：

```
ex:butler    ex:killed    ex:gardener .
```

这是一种“嵌套的”表示，即三元组的宾语也是一个三元组。

另一种建模方法：物化（Reification）



- 做法：引入辅助接点，用于描述要声明的命题中的三元组，该接点用作“句柄”来指代整个声明：

```
ex:detective    ex:supposes    ex:theory .  
ex:theory      rdf:subject    ex:butler .  
ex:theory      rdf:predicate  ex:hasKilled .  
ex:theory      rdf:object     ex:gardener .
```

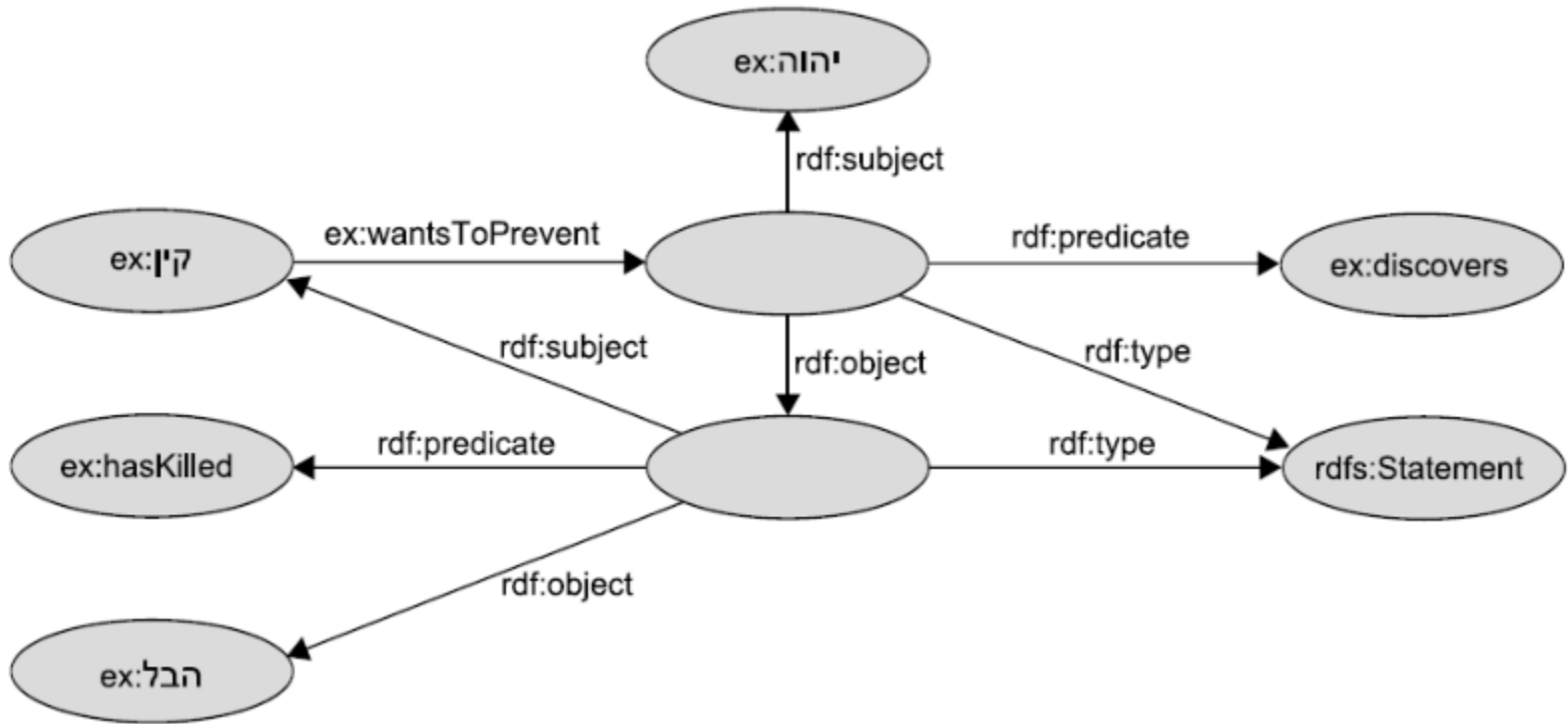
```
ex:theory      rdf:type       rdf:Statement .
```

- 以上声明不能推理出下面的逻辑结论：

```
ex:butler      ex:hasKilled    ex:gardener .
```

- 如果物化命题只被本身引用，可以用空白接点来表示，代替`ex:theory`.

一个物化实例



- 用空接点代替`ex:theory`.



RDF Schema知识点

- 1. 动机
- 2. 类和类层次结构 Classes and Class Hierarchies
- 3. 属性和属性层次结构 Properties and Property Hierarchies
- 4. 属性约束 Property Restrictions
- 5. 开放列表回顾 Open Lists Revisited
- 6. 关于命题的命题：物化 Reification
- 7. **RDFS附加信息 Supplementary Information in RDFS**
- 8. RDFS中的简单本体 Simple Ontologies in RDFS

补充信息



- 注解不是实际本体的部分，但是方便读者、用户和开发人员阅读
- 在**RDF**中，也用三元组进行编码
- **Rdf**提供了一些预定义的属性来表示注解。
- **rdfs:label**: URI命名e.g. to give a human-readable name for a URI
- **rdfs:comment**: 用于为资源指定注解
- **rdfs:seeAlso**, **rdfs:definedBy**:用于连接到提供关于主语资源更多信息或定义的URIs.

```
:
xmlns:wikipedia="http://en.wikipedia.org/wiki/"
:
<rdfs:Class rdf:about="&ex;Primates">
  <rdfs:label xml:lang="en">primates</rdfs:label>
  <rdfs:comment>
    Order of mammals. Primates are characterized by an
    advanced brain. They mostly populate the tropical
    earth regions. The term 'Primates' was coined by
    Carl von Linné.
  </rdfs:comment>
  <rdfs:seeAlso rdf:resource="&wikipedia;Primates" />
  <rdfs:subClassOf rdf:resource="&ex;Mammalia" />
</rdfs:Class>
```



RDF Schema知识点

- 1. 动机
- 2. 类和类层次结构 Classes and Class Hierarchies
- 3. 属性和属性层次结构 Properties and Property Hierarchies
- 4. 属性约束 Property Restrictions
- 5. 开放列表回顾 Open Lists Revisited
- 6. 关于命题的命题：物化 Reification
- 7. RDFS附加信息 Supplementary Information in RDFS
- 8. RDFS中的简单本体 Simple Ontologies in RDFS

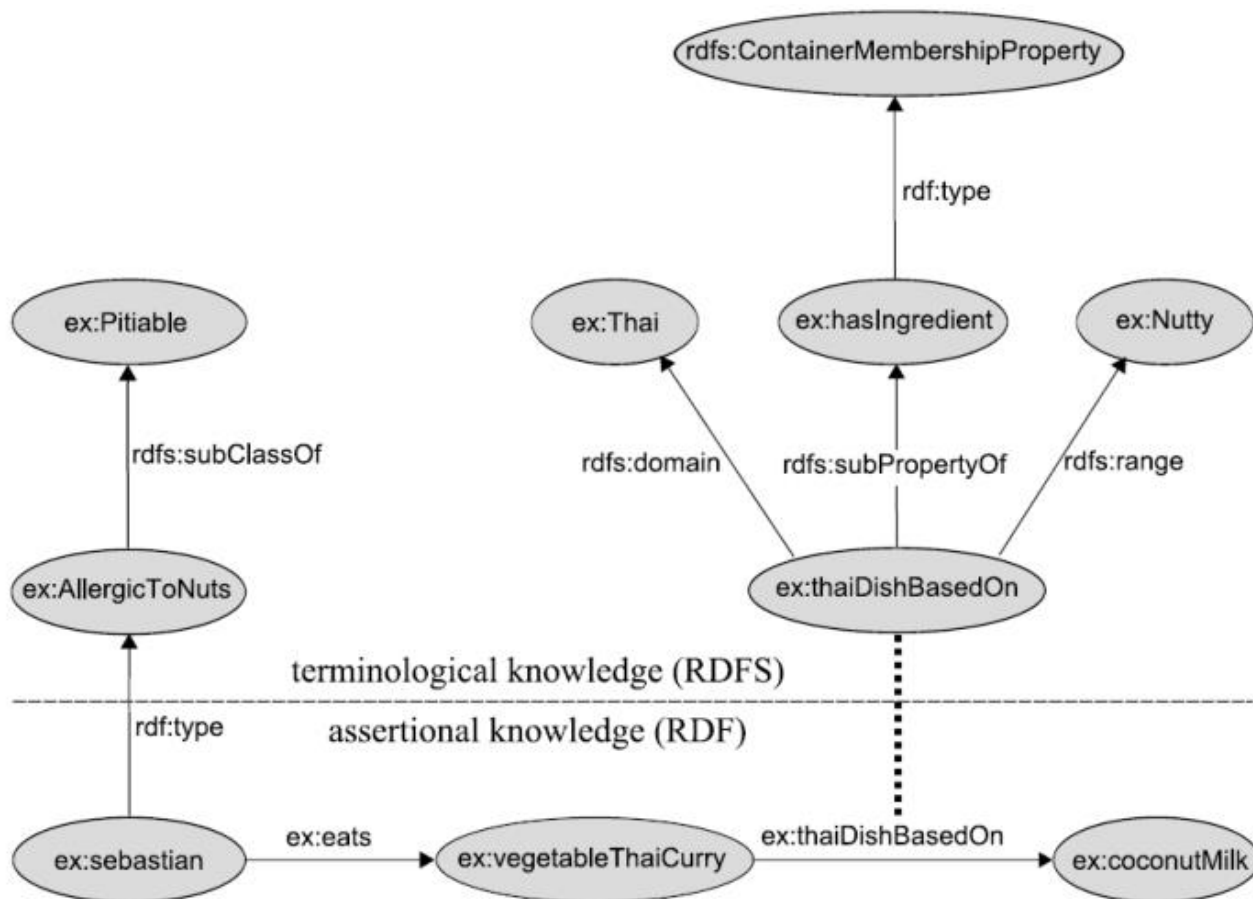
一个本体实例



```
ex:vegetableThaiCurry    ex:thaiDishBasedOn    ex:coconutMilk .
ex:sebastian              rdf:type                ex:AllergicToNuts .
ex:sebastian              ex:eats                    ex:vegetableThaiCurry .

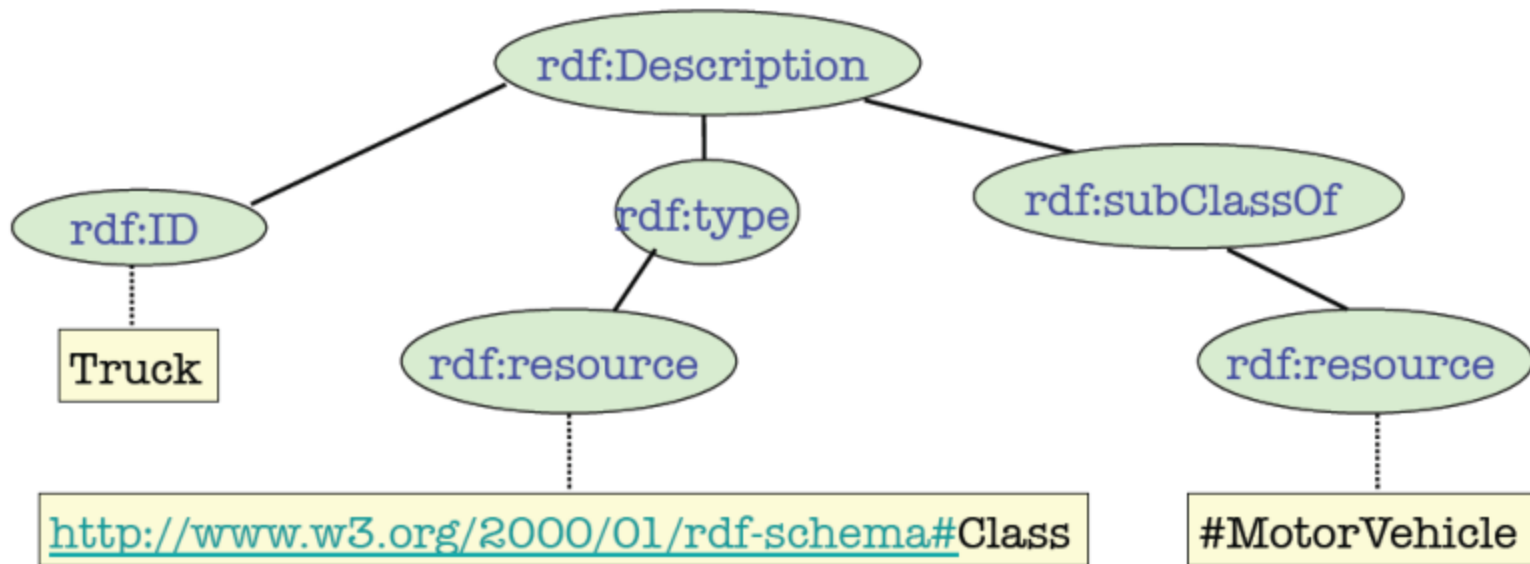
ex:AllergicToNuts         rdfs:subClassOf        ex:Pitiabale .
ex:thaiDishBasedOn        rdfs:domain            ex:Thai .
ex:thaiDishBasedOn        rdfs:range              ex:Nutty .
ex:thaiDishBasedOn        rdfs:subPropertyOf      ex:hasIngredient .
ex:hasIngredient          rdf:type                rdfs:ContainerMembershipProperty.
```


一个本体实例



注意XML的多视图

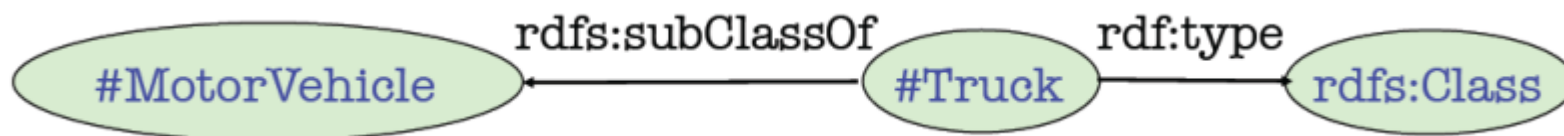
```
<rdf:Description rdf:ID="Truck">  
  <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>  
  <rdfs:subClassOf rdf:resource="#MotorVehicle"/>  
</rdf:Description>
```



注意XML的多视图

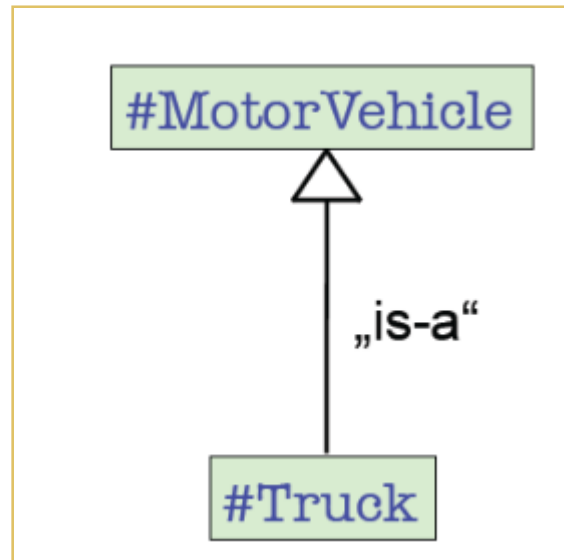


```
<rdf:Description rdf:ID="Truck">  
  <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>  
  <rdfs:subClassOf rdf:resource="#MotorVehicle"/>  
</rdf:Description>
```



注意XML的多视图

```
<rdf:Description rdf:ID="Truck">  
  <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>  
  <rdfs:subClassOf rdf:resource="#MotorVehicle"/>  
</rdf:Description>
```



一、RDFS

二、RDF(S)形式语义



回顾: 隐式知识 Implicit knowledge

- if an RDFS document contains

```
u    rdf:type    ex:Textbook .
```

- and

```
ex:Textbook    rdfs:subClassOf    ex:Book .
```

- Then

```
u    rdf:type    ex:Book .
```

这是一个隐式知识的案例，也是获得逻辑结论（logical consequence）的过程，我们称之为演绎(deduction)或推理(inference)，我们不需要显式地声明。

什么样的陈述（statement）是逻辑结论，由形式语义决定。

回顾: 隐式知识 Implicit knowledge



- from

```
ex:Textbook    rdfs:subClassOf    ex:Book .
```

```
ex:Book        rdfs:subClassOf    ex:PrintMedia .
```

- 得到以下逻辑结论

```
ex:Textbook    rdfs:subClassOf    ex:PrintMedia .
```

也就是说rdfs:subClassOf这个属性是传递的。



为什么定义语义

- “语义”经常出现在不同的背景（逻辑、语言、程序语言等），最恰当表示就是“含义”。
- 我们主要考虑语义概念的**逻辑维度(logic dimension)**，即形式语义(formal semantics)
- 引入RDF(S)语义是必要的，因为从给定的说明中得到结论的解释还没有得到充分讨论。
- 虽然有些特定情况，可以通过样例来获得结论是否有效，但不能保证对于无限多种推理，都能达成共识。解决这一问题最好的方法就是通过定义良好的形式语义来避免非形式化规格说明造成的不确定性。
- 对于RDF(S)来说，命题就是三元组。



为什么定义语义

- 给定一个逻辑，用 \mathbb{P} 表示命题的全集。
- 用一个符号 \models 表示推论关系 (entailment relation) ,比如命题 p_3 和 p_4 是命题 p_1 和 p_2 的逻辑推论，表示为 $\{p_1, p_2\} \models \{p_3, p_4\}$ 。
- 推论关系把命题的集合与命题的集合关联起来 (即 $\models \subseteq 2^{\mathbb{P}} \times 2^{\mathbb{P}}$)
- 一个逻辑 L 是由一个命题集合一个推论关系共同组成的，在抽象的层次上可以被描述为 $L=(\mathbb{P}, \models)$ 。

RDF(S)语义



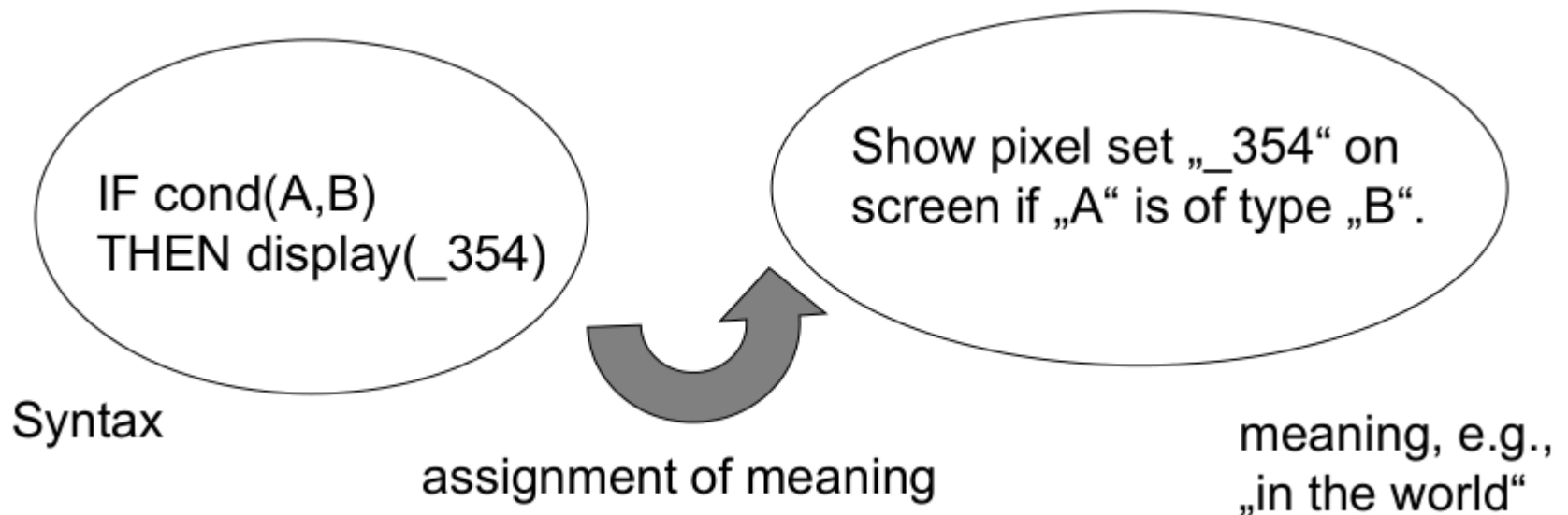
- **1. 什么是语义**What is Semantics?
- **2. 什么是模型论语义**What is Model-theoretic Semantics?
- **3. RDF(S)模型论语义**Model-theoretic Semantics for RDF(S)
- **4. 什么是证明论语义**What is Proof-theoretic Semantics?
- **5. RDF(S)的证明论语义**Proof-theoretic Semantics for RDF(S)

语法和语义

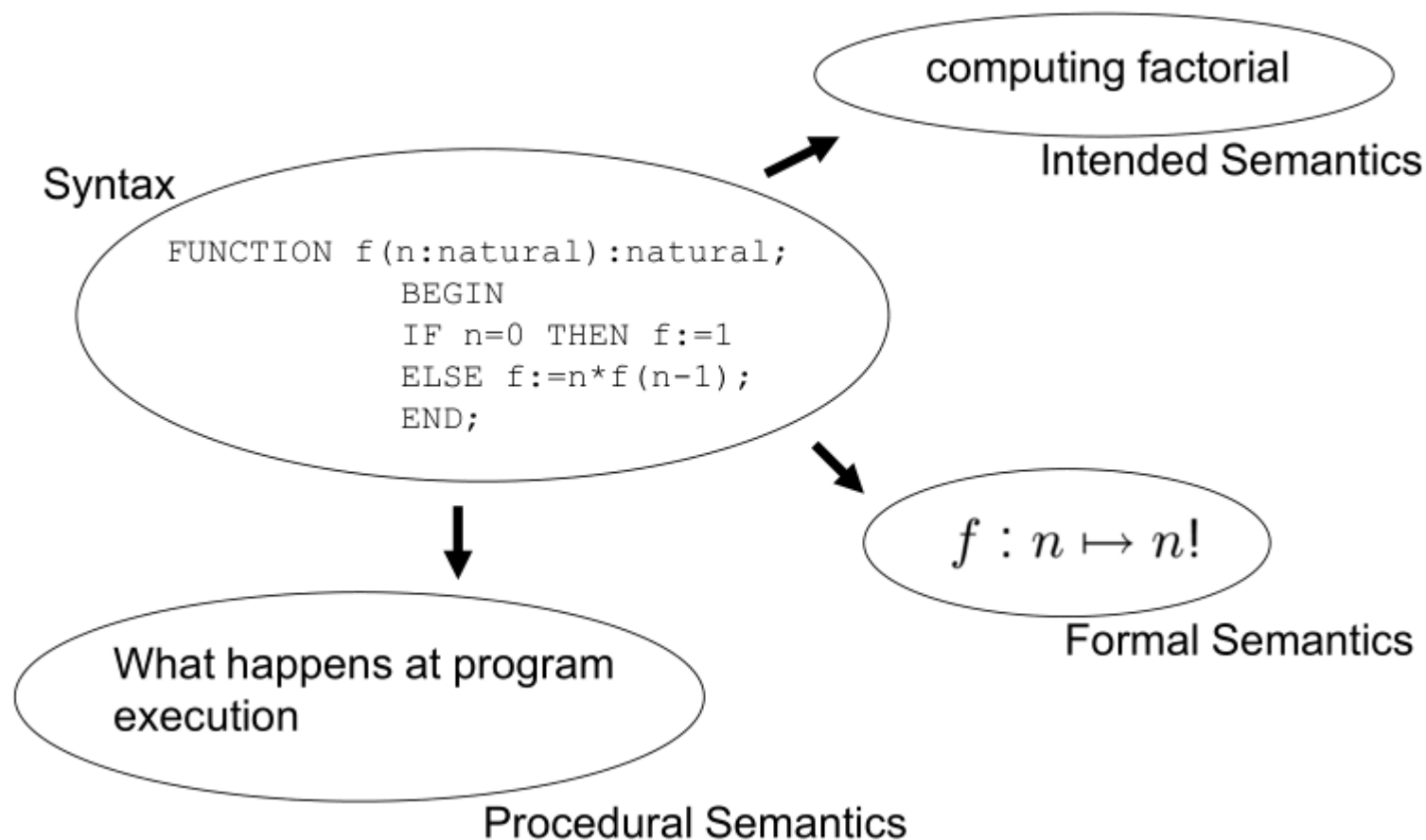


Syntax: 没有“意义”的字符串 **character strings without meaning**

Semantics: 有“意义”的字符串 **meaning of the character strings**



程序语言(Programming Language)的语义



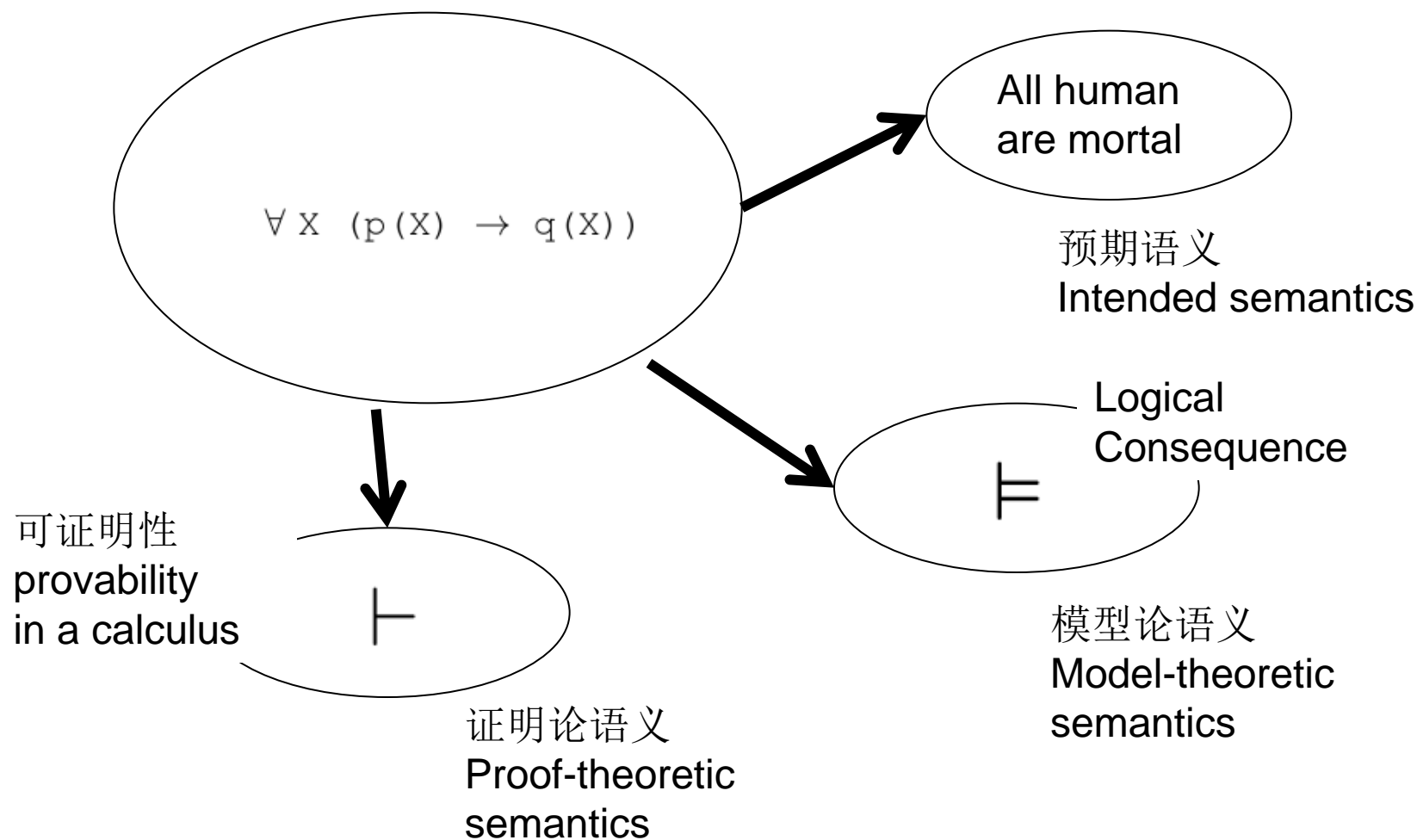
逻辑语义(Semantic of Logic)



- 逻辑语义主要包括两个方面：模型论和证明论。
 - 模型论给出程序的一个模型，说明性地给出程序的意义。
 - 证明论给出一个证明过程，得到程序所能推导出的逻辑结论。
- 一般而言，模型论语义的研究是从一些逻辑背景出发，定义程序的模型，并使其接近一般推导的“直观感觉”。
- 证明论是以模型论为基础的。它给出一个证明过程，其正确性和完备性是基于某种模型论的。
- 另一方面，证明论是实现模型论的基础，有的证明论语义可以直接加以实现，但有的可能涉及到无限推导或算法复杂度过高，难以实现。

参考文献：《综述:一般逻辑程序的证明论语义》，计算机科学，2004

逻辑语义(Semantic of Logic)





什么样的语义是好的

- 语义网需要一种可共享，陈述性和可计算语义。 Semantic Web requires a shareable, declarative and computable semantics.
- 也就是说，语义对象（实体）必须清晰定义，以及可自动计算。 The semantics must be a formal entity which is clearly defined and automatically computable.
- 本体语言通过他们的形式语义提供上述所说的语义表示。 Ontology languages provide this by means of their formal semantics.
- 语义网的语义通过**逻辑结论（关系）形式**进行表示。 Semantic Web Semantics is given by a relation – **the logical consequence relation**.

换句话说



- 我们获取语义信息不是通过指定其意思（含义meaning）（这是不可能的）
- 而是通过定义信息如何与其他信息进行交互，并通过信息交互产生的结果来非直接地描述它的语义。



RDF(S)语义

- 1. 什么是语义 What is Semantics?
- 2. 什么是模型论语义 What is Model-theoretic Semantics?
- 3. RDF(S)模型论语义 Model-theoretic Semantics for RDF(S)
- 4. 什么是证明论语义 What is Proof-theoretic Semantics?
- 5. RDF(S)的证明论语义 Proof-theoretic Semantics for RDF(S)



Interpretation解释

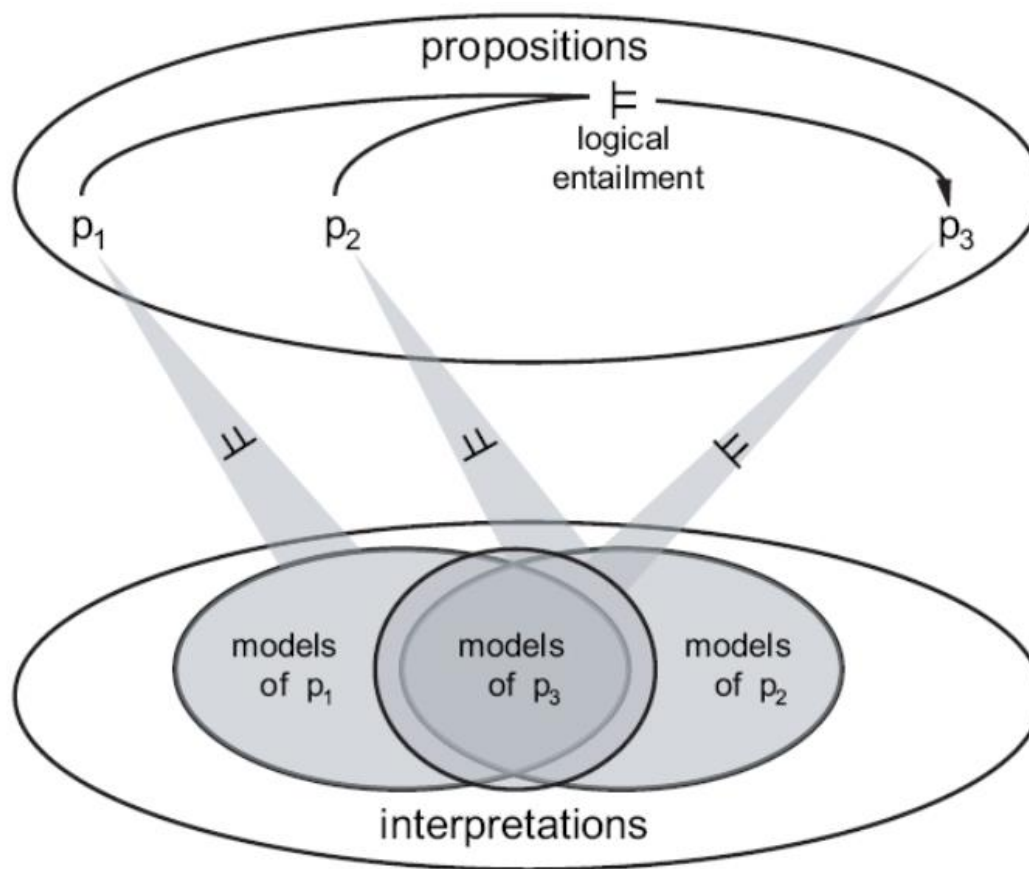
- Interpretation是模型论的核心概念，它可以理解为潜在的“现实”或“世界”，尤其是它不必遵从实际的现实。
- 在形式逻辑中，通常选择特定的数学结构作为解释，以使之以形式化正确的方式起作用。选择哪种结构取决于所考虑的逻辑。
- 如何决定一个特定的解释 I 是否满足一个特定的逻辑命题（在这里，我们称 I 是 p 的模型，记作 $I \models p$ ，与推论关系使用相同的符号）。
- 此外，对于命题集合 $P \subseteq \mathbb{P}$ ，如果 I 是 P 中每个 p 的模型，我们说 I 是 P 的一个模型（记作 $I \models P$ ）。
- 一个命题集 $P' \subseteq \mathbb{P}$ 是由一个命题集 $P \subseteq \mathbb{P}$ 推理得出的，当且仅当每个满足 P 中所有命题 p 的模型（ $I \models P$ ）都是 P' 中每个命题 p' 的模型（ $I \models P'$ ）。

Interpretation解释实例



- 两个命题，light_green, green
- 解释 I ：真实世界的东西
- 因为每个对象（解释，真实世界的东西）满足light_green(即， $I \models \text{light_green}$)，也自动成为一个命题green的模型，即 $I \models \text{green}$ ，因此 $\{\text{light_green}\} \models \text{green}$
- 所以： I 是 $\text{light_green} \models \text{green}$ 的一个模型（解释）

逻辑推论 (Logical Consequence)



P1和p2命题推出p3,则p3的模型是p1和p2模型的交集。

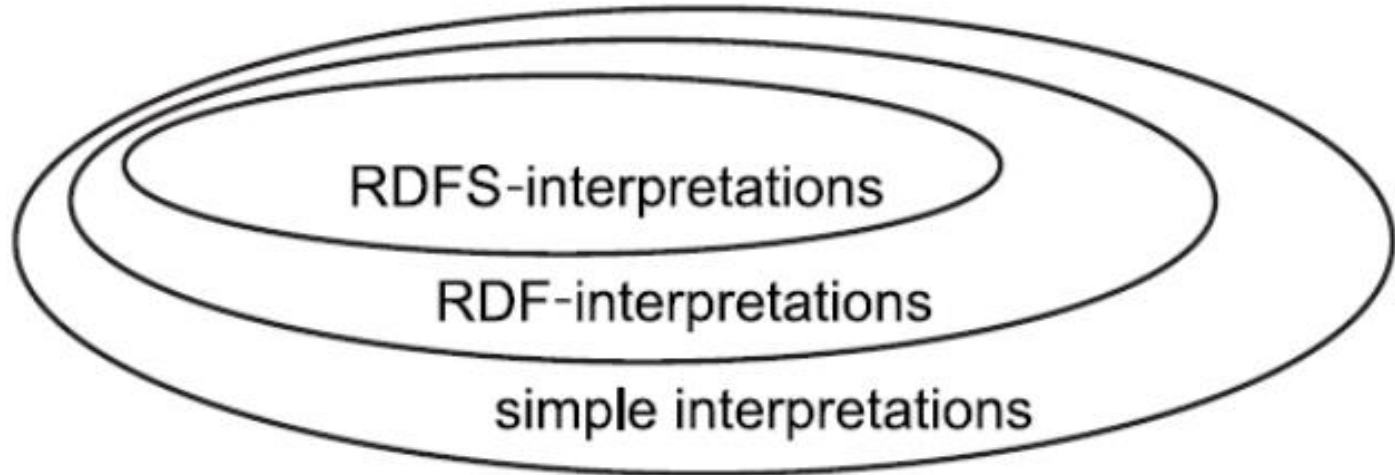


RDF(S)语义

- 1. 什么是语义What is Semantics?
- 2. 什么是模型论语义What is Model-theoretic Semantics?
- 3. **RDF(S)模型论语义Model-theoretic Semantics for RDF(S)**
- 4. 什么是证明论语义What is Proof-theoretic Semantics?
- 5. RDF(S)的证明论语义Proof-theoretic Semantics for RDF(S)

RDF(S)的模型语义

- 语言：任何有效的RDF(S)，命题就是三元组（图就是三元组的集合）
- 解释器由函数和集合给出，就是通过函数将语言中的词汇映射到集合中的三元组
- 通过获取RDF(S)术语的含义来定义模型。
- RDF(s)模型语义分为三层：



Simple Interpretations



So we define: a *simple interpretation* \mathcal{I} of a given vocabulary V consists of

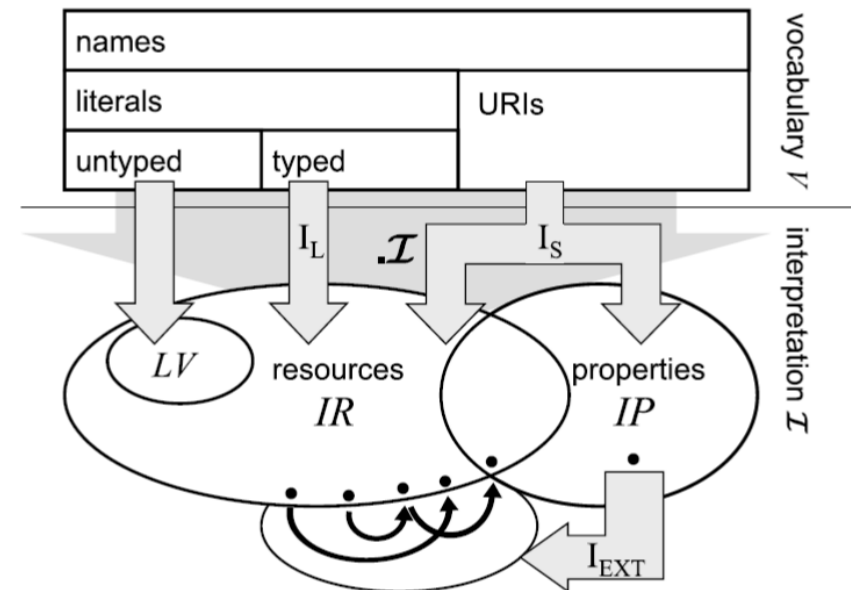
- IR , a non-empty set of *resources*, alternatively called domain or universe of discourse of \mathcal{I} , 也称领域, 或者 I 的论域, 即资源的模型
- IP , the set of *properties* of \mathcal{I} (which may overlap with IR), 属性的模型
- I_{EXT} , a function assigning to each property a set of pairs from IR , i.e. $I_{EXT} : IP \rightarrow 2^{IR \times IR}$, where $I_{EXT}(p)$ is called the *extension* of the property p , 一个函数, 为每个属性指定一个 IR 中元素对的集合, 属性(关系)扩展模型
- I_S , a function, mapping URIs from V into the union of the sets IR and IP , i.e. $I_S : V \rightarrow IR \cup IP$, 一个函数, 把 V 中的URI映射到 IR 和 IP 的并集上。
- I_L , a function from the typed literals from V into the set IR of resources and 一个函数, 把 V 中的有类型字面体值映射到 IR 资源集
- LV , a particular subset of IR , called the set of *literal values*, containing (at least) all untyped literals from V .

LV 是 IR 的一个特殊子集, 称为字面量值, 包括(至少) V 中所有无类型的字面量。

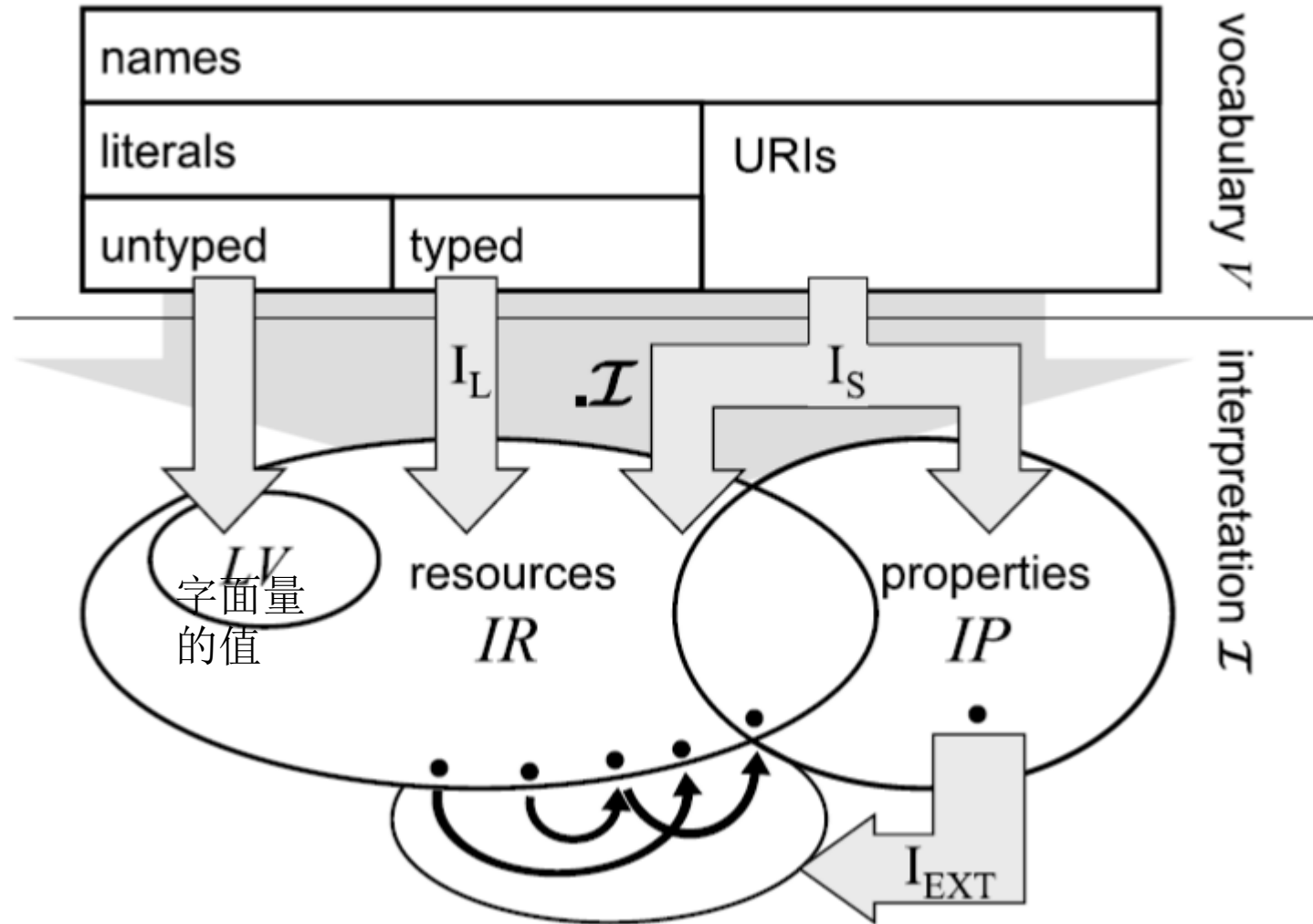
Simple Interpretations

基于集合 IR 、 IP 、 LV 和函数 I_{EXT} 、 I_S 和 I_L ，定义解释函数 \cdot^I ，把词汇表 V 中的所有 URI 与字面量映射为资源和属性。

- every untyped literal " a " is mapped to a , formally: $(\text{"a"})^I = a$,
- every untyped literal carrying language information " a "@ t is mapped to the pair $\langle a, t \rangle$, i.e. $(\text{"a"}@t)^I = \langle a, t \rangle$,
- every typed literal l is mapped to $I_L(l)$, formally: $l^I = I_L(l)$, and
- every URI u is mapped to $I_S(u)$, i.e. $u^I = I_S(u)$.



Simple Interpretations



三元组解释案例



我们考虑图表 2.7 的图作为例子。对应的词汇表 V 由该图的结点和边的名称构成。我们给出这个词汇的一个简单解释 I 如下：

$$\begin{aligned}
 IR &= \{\chi, v, \tau, V, \epsilon, l, 1lb\} & I_S &= \begin{aligned} &ex:chutney \mapsto \chi \\ &ex:greenMango \mapsto v \\ &ex:hasIngredient \mapsto \tau \\ &ex:ingredient \mapsto V \\ &ex:amount \mapsto l \end{aligned} \\
 IP &= \{\tau, V, l\} \\
 LV &= \{1lb\} \\
 I_{EXT} &= \begin{aligned} &\tau \mapsto \{\langle \chi, \epsilon \rangle\} \\ &V \mapsto \{\langle \epsilon, v \rangle\} \\ &l \mapsto \{\langle \epsilon, 1lb \rangle\} \end{aligned} & I_L &= \text{是“空函数”，因为没有有类型的字面量}
 \end{aligned}$$

给定 $A: _ : id1 \mapsto \epsilon$, 注意 $I+A$ 将我们考虑的图的三个三元组都赋值为真：

$$\begin{aligned}
 \langle ex:chutney^{I+A}, _ : id1^{I+A} \rangle &= \langle \chi, \epsilon \rangle \in I_{EXT}(\tau) = I_{EXT}(ex:hasIngredient^{I+A}) \\
 \langle _ : id1^{I+A}, ex:greenMango^{I+A} \rangle &= \langle \epsilon, v \rangle \in I_{EXT}(V) = I_{EXT}(ex:ingredient^{I+A}) \\
 \langle _ : id1^{I+A}, "1lb"^{I+A} \rangle &= \langle \epsilon, 1lb \rangle \in I_{EXT}(l) = I_{EXT}(ex:amount^{I+A})
 \end{aligned}$$

因此，该图也被赋值为真。所以，简单解释 I 是该图的一个模型。

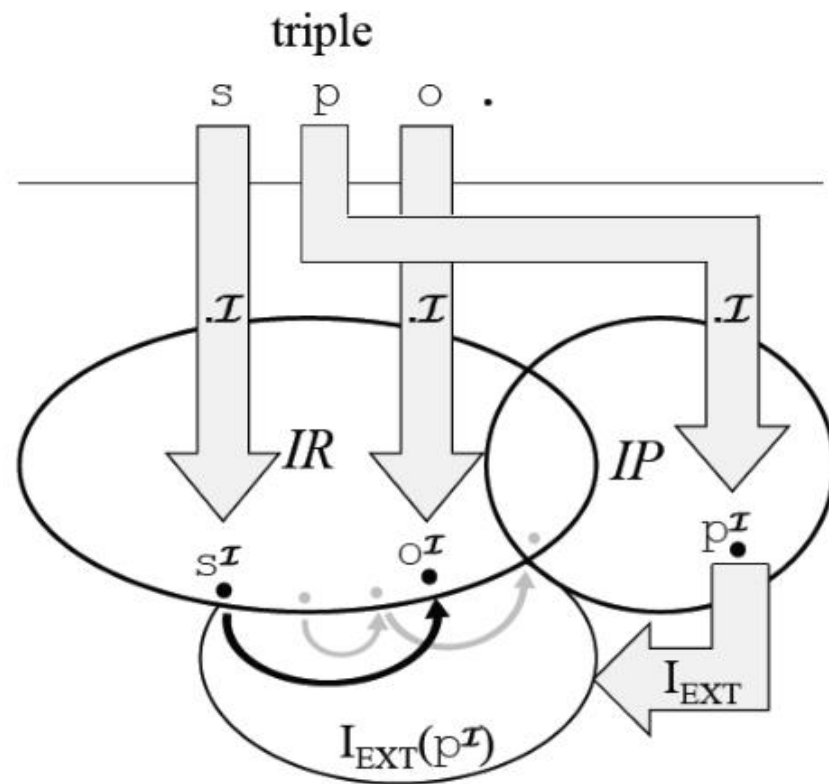
简单解释 (Simple Interpretations)



基础三元组 $s \ p \ o.$ 的真值 $s \ p \ o.^{\mathcal{I}}$ 为真, 当且仅当他的所有组成成分 **S**、**P**、**O** 都在词汇表 **V** 中, 并且 $\langle s^{\mathcal{I}}, o^{\mathcal{I}} \rangle \in I_{\text{EXT}}(p^{\mathcal{I}})$. 成立。

直观地说, 后面的条件要求从分配给 **S** 与 **O** 的资源中构造的对是在 **p** 指定的属性扩展中。

注: 基础三元组不包含空节点



I_{EXT} 为每个属性指定一个 **IR** 中元素对的集合

简单蕴含 (Simple entailment)



解释函数. I 对每个基础图 G 指定了真值 : G^I 是真值 , 当且仅当图 G 中的每个三元组为真。

如果每个解释是 G 的模型 , 同时也是 G' 的模型。则图 G 简单蕴含了一个图 G'

三元组解释案例



我们考虑图表 2.7 的图作为例子。对应的词汇表 V 由该图的结点和边的名称构成。我们给出这个词汇的一个简单解释 I 如下：

IR	$= \{\chi, v, \tau, V, \epsilon, l, lb\}$	I_S	$ex:chutney \mapsto \chi$
IP	$= \{\tau, V, l\}$		$ex:greenMango \mapsto v$
LV	$= \{lb\}$		$ex:hasIngredient \mapsto \tau$
I_{EXT}	$\tau \mapsto \{\langle \chi, \epsilon \rangle\}$		$ex:ingredient \mapsto V$
	$V \mapsto \{\langle \epsilon, v \rangle\}$	I_L	是“空函数”，因为
	$l \mapsto \{\langle \epsilon, lb \rangle\}$		没有有类型的字面量

给定 $A : _ : id1 \mapsto \epsilon$, 注意 $I+A$ 将我们考虑的图的三个三元组都赋值为真：

$$\begin{aligned} \langle ex:chutney^{I+A}, _ : id1^{I+A} \rangle &= \langle \chi, \epsilon \rangle \in I_{EXT}(\tau) = I_{EXT}(ex:hasIngredient^{I+A}) \\ \langle _ : id1^{I+A}, ex:greenMango^{I+A} \rangle &= \langle \epsilon, v \rangle \in I_{EXT}(V) = I_{EXT}(ex:ingredient^{I+A}) \\ \langle _ : id1^{I+A}, "1lb"^{I+A} \rangle &= \langle \epsilon, lb \rangle \in I_{EXT}(l) = I_{EXT}(ex:amount^{I+A}) \end{aligned}$$

因此，该图也被赋值为真。所以，简单解释 I 是该图的一个模型。

当三元组图包含空白节点时。如果可以把每一个空白节点替换为一个资源，使得替换后不包含空白节点的图都变得有效，那么原图就是有效的。

设 A 是一个把 IR 中资源指定给空白节点的函数。
定义一个映射 A 和一个解释 I 组合的解释 $I+A$ 。



RDF-解释 Part 1

一个基于词汇表 V 的RDF-解释是基于词汇表的简单解释，并且满足下面附加条件：

- $x \in IP$ exactly if $\langle x, \text{rdf:Property}^I \rangle \in I_{\text{EXT}}(\text{rdf:type}^I)$.
- if " s "^{^^}`rdf:XMLLiteral` is contained in V and s is a well-typed XML-Literal, then
 - $I_L("s"^{^^}\text{rdf:XMLLiteral})$ is the XML value of s ;

词汇表：

`rdf:type` `rdf:Property` `rdf:XMLLiteral` `rdf:nil` `rdf:List` `rdf:Statement` `Rdf:subject`
`rdf:predicate` `rdf:object` `rdf:first` `rdf:rest:seq` `rdf:Bag` `rdf:Alt` `rdf:value`

- if " s "^{^^}`rdf:XMLLiteral` is contained in V and s is an ill-typed XML literal, then
 - $I_L("s"^{^^}\text{rdf:XMLLiteral}) \notin LV$ and
 - $\langle I_L("s"^{^^}\text{rdf:XMLLiteral}), \text{rdf:XMLLiteral}^I \rangle \notin I_{\text{EXT}}(\text{rdf:type}^I)$.

RDF-解释 Part 2



除了前面的限制以外，RDF-解释必须满足以下所有所有三元组（公理三元组）为真的条件：

<code>rdf:type</code>	<code>rdf:type</code>	<code>rdf:Property.</code>
<code>rdf:subject</code>	<code>rdf:type</code>	<code>rdf:Property.</code>
<code>rdf:predicate</code>	<code>rdf:type</code>	<code>rdf:Property.</code>
<code>rdf:object</code>	<code>rdf:type</code>	<code>rdf:Property.</code>
<code>rdf:first</code>	<code>rdf:type</code>	<code>rdf:Property.</code>
<code>rdf:rest</code>	<code>rdf:type</code>	<code>rdf:Property.</code>
<code>rdf:value</code>	<code>rdf:type</code>	<code>rdf:Property.</code>
<code>rdf:_<i>i</i></code>	<code>rdf:type</code>	<code>rdf:Property.</code>
<code>rdf:nil</code>	<code>rdf:type</code>	<code>rdf:List.</code>



RDFS-解释 Part 1

- 定义：给定的RDFS解释. I ，它可以把资源映射为资源的集合，形式化表

$$I_{\text{CEXT}} : IR \rightarrow 2^{IR}$$

我们定义 $I_{\text{CEXT}}(y)$ 为形如 $\langle x, y \rangle$ 被包含在 $I_{\text{EXT}}(\text{rdf:type}^I)$ 中的元素 x 的集合， $I_{\text{CEXT}}(y)$ 也被称为 y 的（类）扩展。

$$- IC = I_{\text{CEXT}}(\text{rdfs:Class}^I).$$

词汇表：

`rdfs:domain` `rdfs:range` `rdfs:Resource` `rdfs:Literal` `rdfs:Datatype` `rdfs:Class`
`rdfs:subClassOf` `rdfs:subPropertyOf` `rdfs:member` `rdfs:Container` `rdfs:comment`
`rdfs:ContainerMembershipProperty` `rdfs:seeAlso` `rdfs:isDefinedBy` `rdfs:label`

then $u \in I_{\text{CEXT}}(y)$.

- If $\langle x, y \rangle \in I_{\text{EXT}}(\text{rdfs:range}^I)$ and $\langle u, v \rangle \in I_{\text{EXT}}(x)$,
then $v \in I_{\text{CEXT}}(y)$.
- $I_{\text{EXT}}(\text{rdfs:subPropertyOf}^I)$ is reflexive and transitive on IP .

RDFS-解释 Part 2



- If $\langle x, y \rangle \in I_{\text{EXT}}(\text{rdfs:subPropertyOf}^{\mathcal{I}})$,
then $x, y \in IP$ and $I_{\text{EXT}}(x) \subseteq I_{\text{EXT}}(y)$.
- If $x \in IC$,
then $\langle x, \text{rdfs:Resource}^{\mathcal{I}} \rangle \in I_{\text{EXT}}(\text{rdfs:subClassOf}^{\mathcal{I}})$.
- If $\langle x, y \rangle \in I_{\text{EXT}}(\text{rdfs:subClassOf}^{\mathcal{I}})$,
then $x, y \in IC$ and $I_{\text{CEXT}}(x) \subseteq I_{\text{CEXT}}(y)$.
- $I_{\text{EXT}}(\text{rdfs:subClassOf}^{\mathcal{I}})$ is reflexive and transitive on IC .
- If $x \in I_{\text{CEXT}}(\text{rdfs:ContainerMembershipProperty}^{\mathcal{I}})$,
then $\langle x, \text{rdfs:member}^{\mathcal{I}} \rangle \in I_{\text{EXT}}(\text{rdfs:subPropertyOf}^{\mathcal{I}})$.
- If $x \in I_{\text{CEXT}}(\text{rdfs:Datatype}^{\mathcal{I}})$,
then $\langle x, \text{rdfs:Literal}^{\mathcal{I}} \rangle \in I_{\text{EXT}}(\text{rdfs:subClassOf}^{\mathcal{I}})$

RDFS-解释 Part 3



RDFS-解释必须满足以下所有所有三元组（公理三元组）为真的条件：

<code>rdf:type</code>	<code>rdfs:domain</code>	<code>rdfs:Resource</code> .
<code>rdfs:domain</code>	<code>rdfs:domain</code>	<code>rdf:Property</code> .
<code>rdfs:range</code>	<code>rdfs:domain</code>	<code>rdf:Property</code> .
<code>rdfs:subPropertyOf</code>	<code>rdfs:domain</code>	<code>rdf:Property</code> .
<code>rdfs:subClassOf</code>	<code>rdfs:domain</code>	<code>rdfs:Class</code> .
<code>rdf:subject</code>	<code>rdfs:domain</code>	<code>rdf:Statement</code> .
<code>rdf:predicate</code>	<code>rdfs:domain</code>	<code>rdf:Statement</code> .
<code>rdf:object</code>	<code>rdfs:domain</code>	<code>rdf:Statement</code> .
<code>rdfs:member</code>	<code>rdfs:domain</code>	<code>rdfs:Resource</code> .
<code>rdf:first</code>	<code>rdfs:domain</code>	<code>rdf:List</code> .
<code>rdf:rest</code>	<code>rdfs:domain</code>	<code>rdf:List</code> .
<code>rdfs:seeAlso</code>	<code>rdfs:domain</code>	<code>rdfs:Resource</code> .
<code>rdfs:isDefinedBy</code>	<code>rdfs:domain</code>	<code>rdfs:Resource</code> .

RDFS-解释 Part 4



RDFS-解释必须满足以下所有所有三元组（公理三元组）为真的条件：

<code>rdfs:comment</code>	<code>rdfs:domain</code>	<code>rdfs:Resource</code> .
<code>rdfs:label</code>	<code>rdfs:domain</code>	<code>rdfs:Resource</code> .
<code>rdf:value</code>	<code>rdfs:domain</code>	<code>rdfs:Resource</code> .
<code>rdf:type</code>	<code>rdfs:range</code>	<code>rdfs:Class</code> .
<code>rdfs:domain</code>	<code>rdfs:range</code>	<code>rdfs:Class</code> .
<code>rdfs:range</code>	<code>rdfs:range</code>	<code>rdfs:Class</code> .
<code>rdfs:subPropertyOf</code>	<code>rdfs:range</code>	<code>rdf:Property</code> .
<code>rdfs:subClassOf</code>	<code>rdfs:range</code>	<code>rdfs:Class</code> .
<code>rdf:subject</code>	<code>rdfs:range</code>	<code>rdfs:Resource</code> .
<code>rdf:predicate</code>	<code>rdfs:range</code>	<code>rdfs:Resource</code> .
<code>rdf:object</code>	<code>rdfs:range</code>	<code>rdfs:Resource</code> .
<code>rdfs:member</code>	<code>rdfs:range</code>	<code>rdfs:Resource</code> .
<code>rdf:first</code>	<code>rdfs:range</code>	<code>rdfs:Resource</code> .
<code>rdf:rest</code>	<code>rdfs:range</code>	<code>rdf:List</code> .
<code>rdfs:seeAlso</code>	<code>rdfs:range</code>	<code>rdfs:Resource</code> .
<code>rdfs:isDefinedBy</code>	<code>rdfs:range</code>	<code>rdfs:Resource</code> .
<code>rdfs:comment</code>	<code>rdfs:range</code>	<code>rdfs:Literal</code> .
■ <code>rdfs:label</code>	<code>rdfs:range</code>	<code>rdfs:Literal</code> .
<code>rdf:value</code>	<code>rdfs:range</code>	<code>rdfs:Resource</code> .

RDFS-解释 Part 5



RDFS-解释必须满足以下所有所有三元组（公理三元组）为真的条件：

```
rdfs:ContainerMembershipProperty
    rdfs:subClassOf      rdf:Property .
rdf:Alt                 rdfs:subClassOf      rdfs:Container
rdf:Bag                 rdfs:subClassOf      rdfs:Container
rdf:Seq                 rdfs:subClassOf      rdfs:Container

rdfs:isDefinedBy        rdfs:subPropertyOf  rdfs:seeAlso .

rdf:XMLLiteral          rdf:type          rdfs:Datatype .
rdf:XMLLiteral          rdfs:subClassOf      rdfs:Literal .
rdfs:Datatype           rdfs:subClassOf      rdfs:Class .

rdf:_i                  rdf:type
                        rdfs:ContainerMembershipProperty

rdf:_i                  rdfs:domain          rdfs:Resource .
rdf:_i                  rdfs:range          rdfs:Resource .
```

小结



- **RDFS语法，RDFS是最简单的本体描述语言**
 - **RDF(S)语义，模型语义概念**
 - **实验：掌握RDFS语言的基本语法，使用RDFS进行本体建模**
-