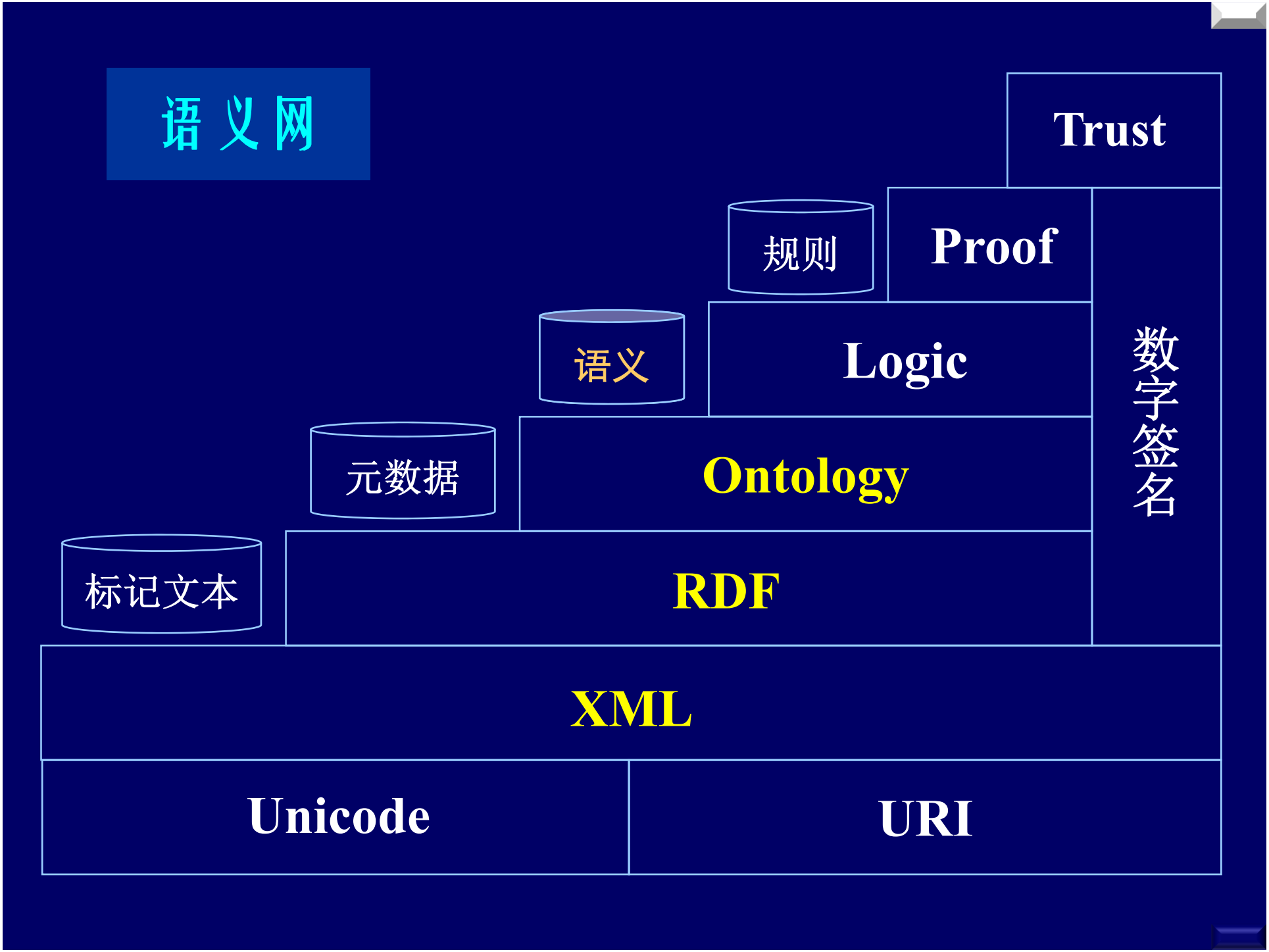
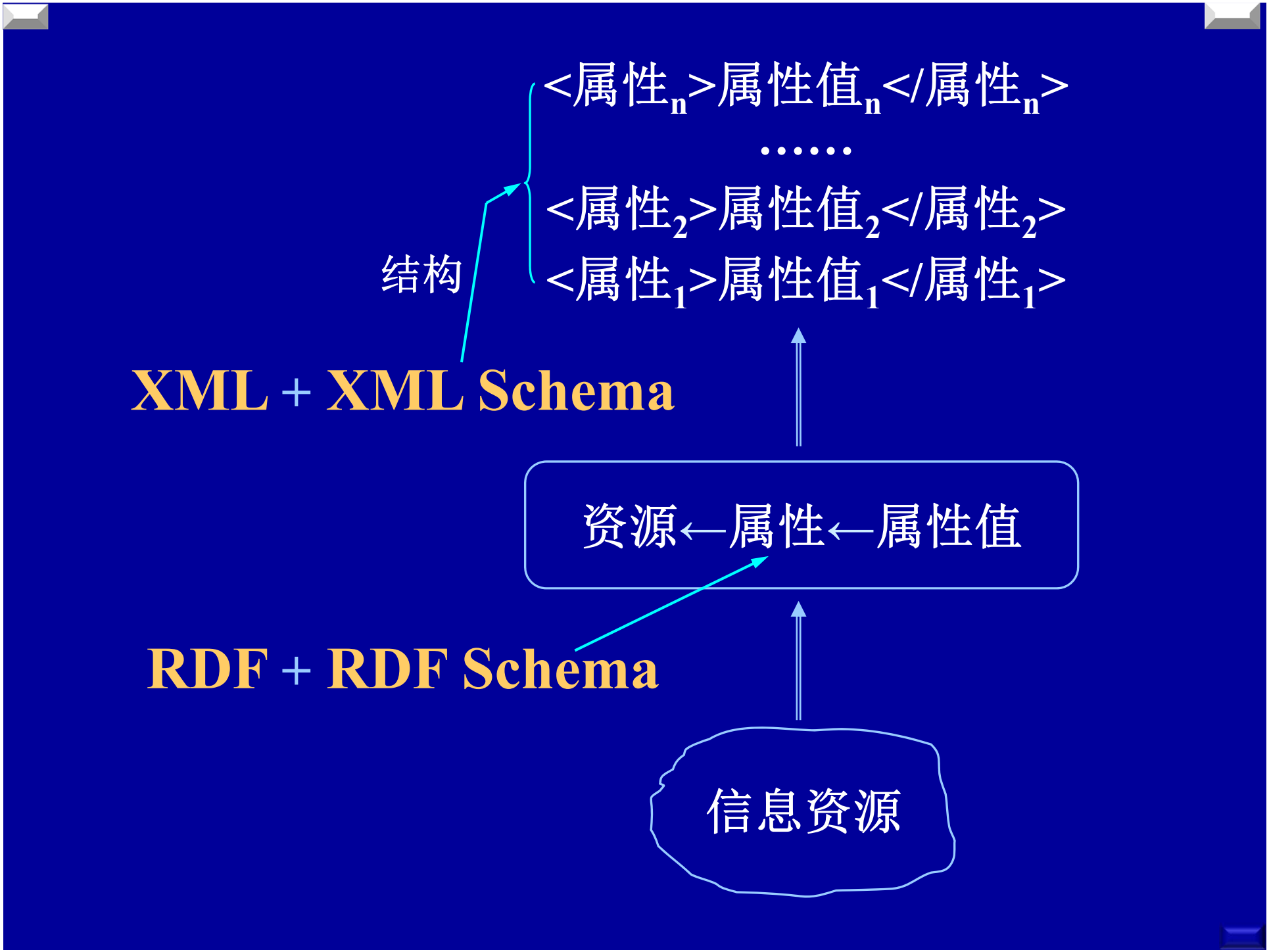
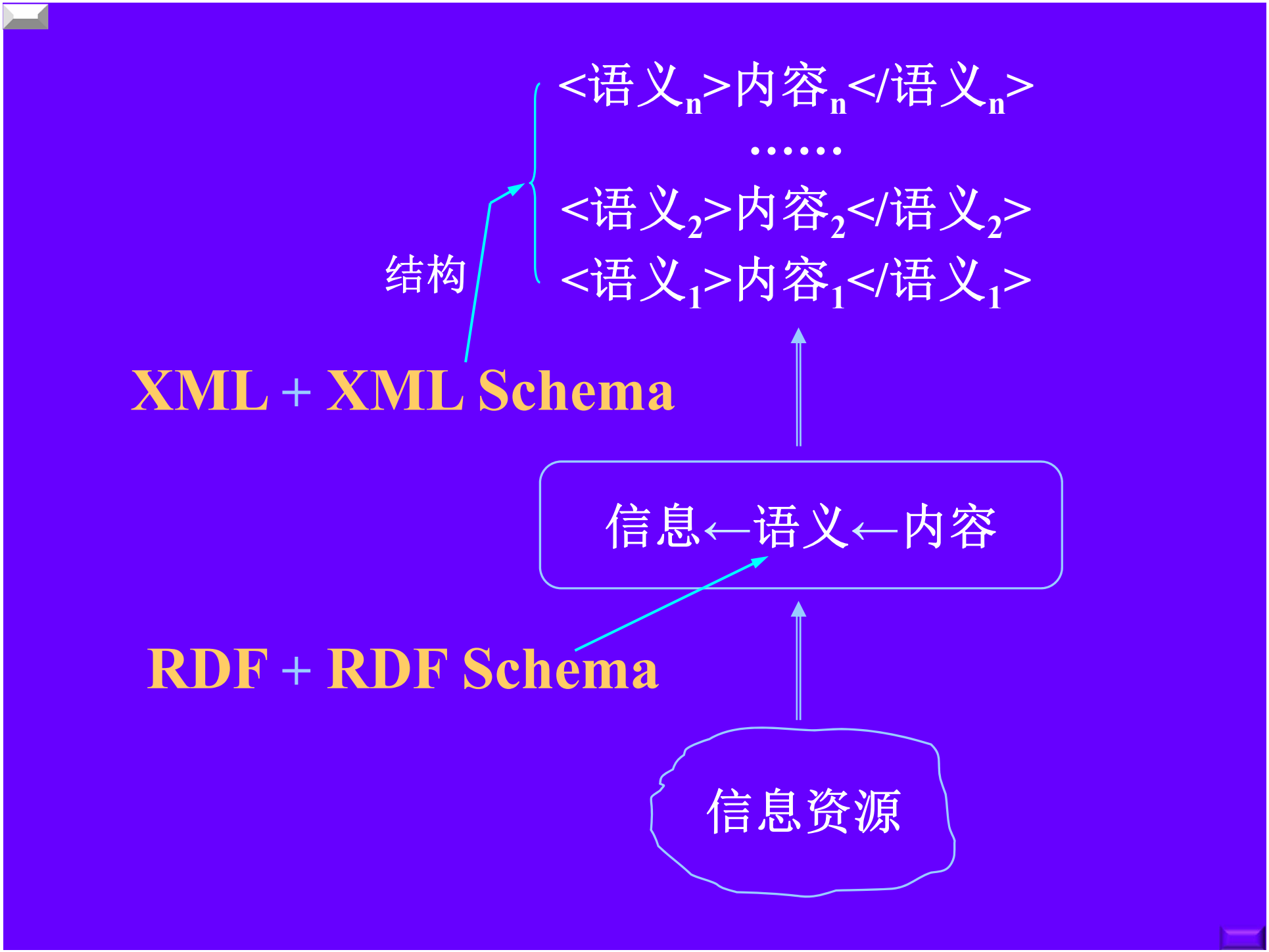
硕士研究生课程《智能信息处理》

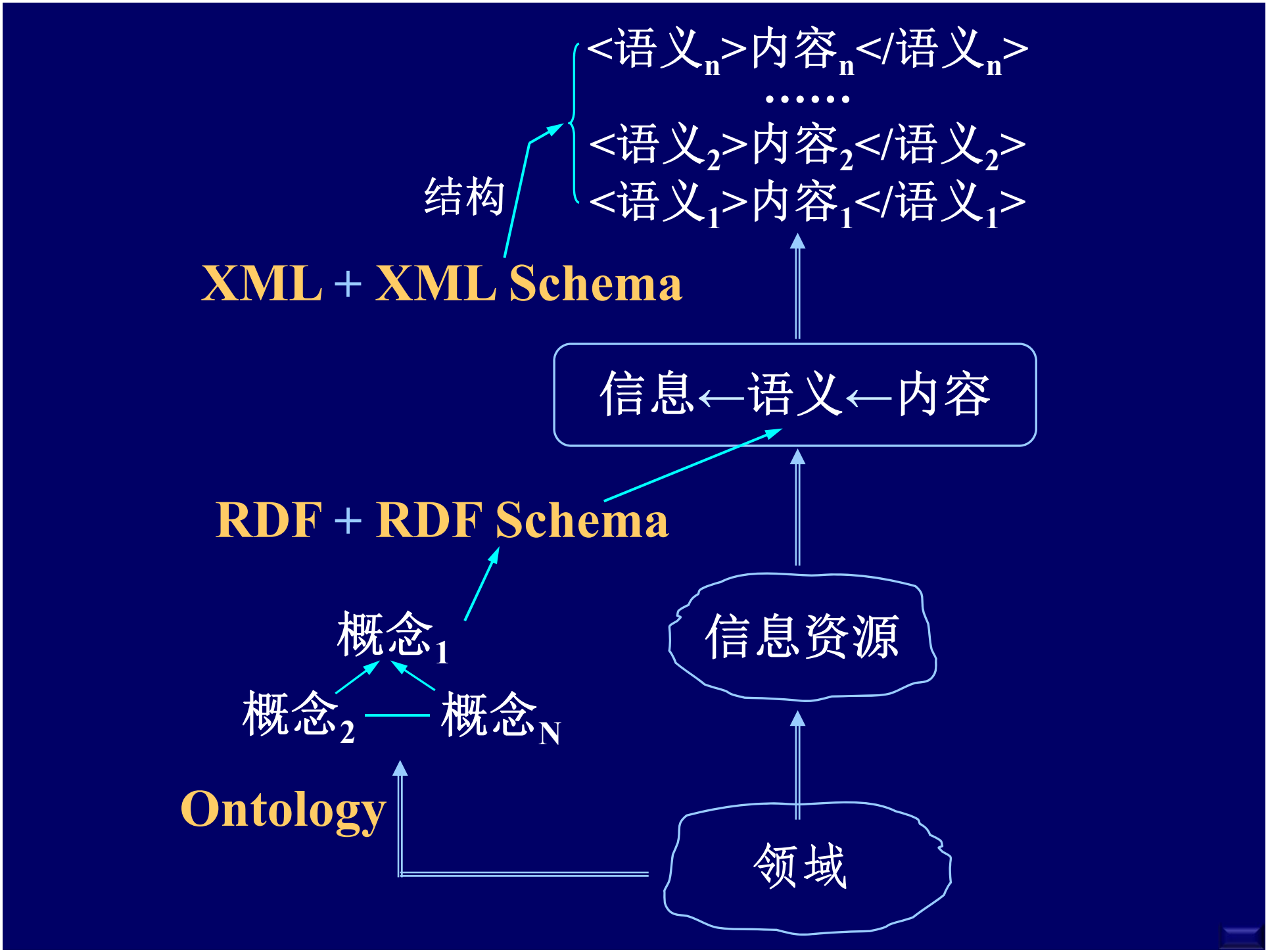
语义网基础理论

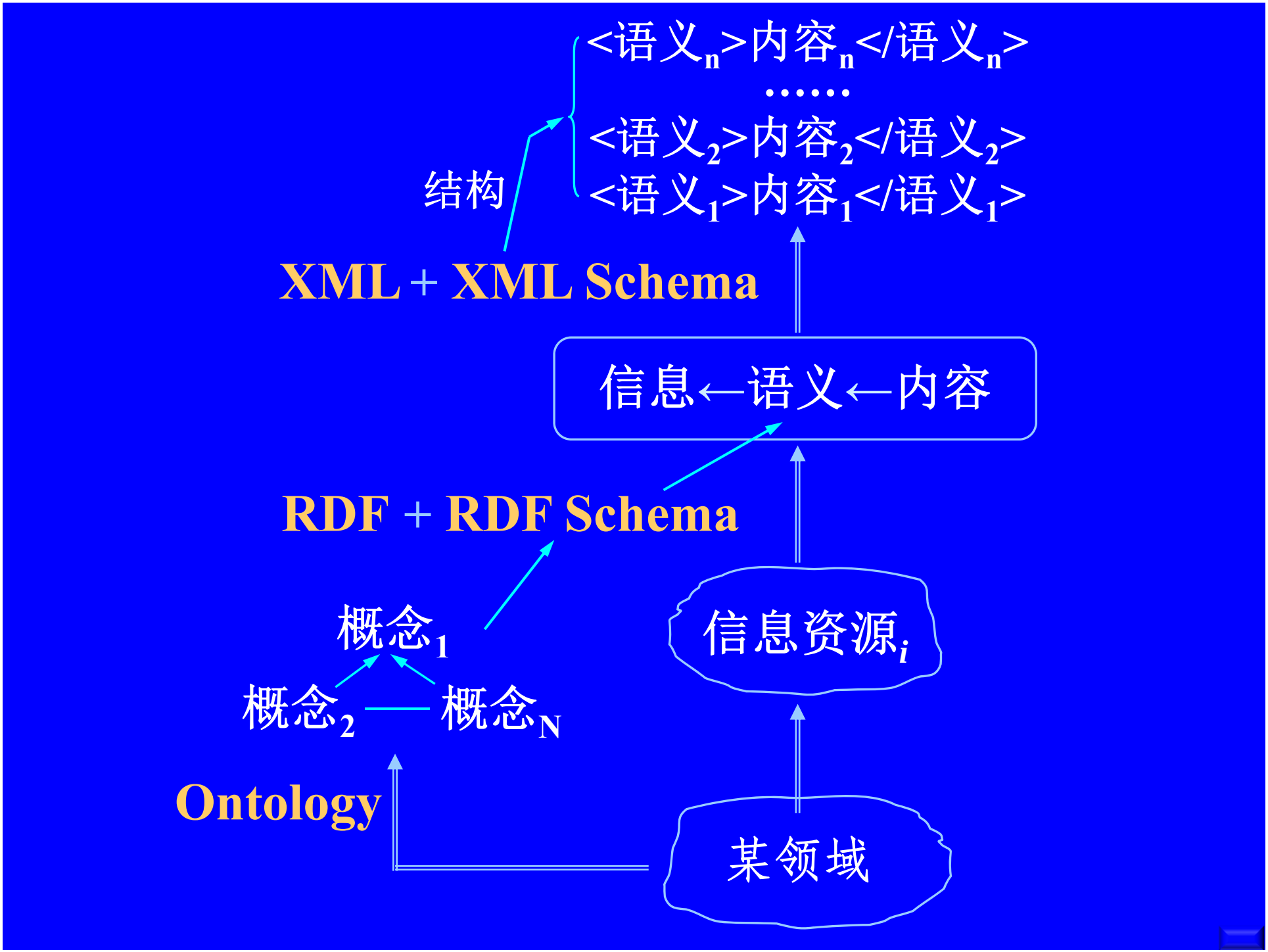
大连海事大学信息科学技术学院

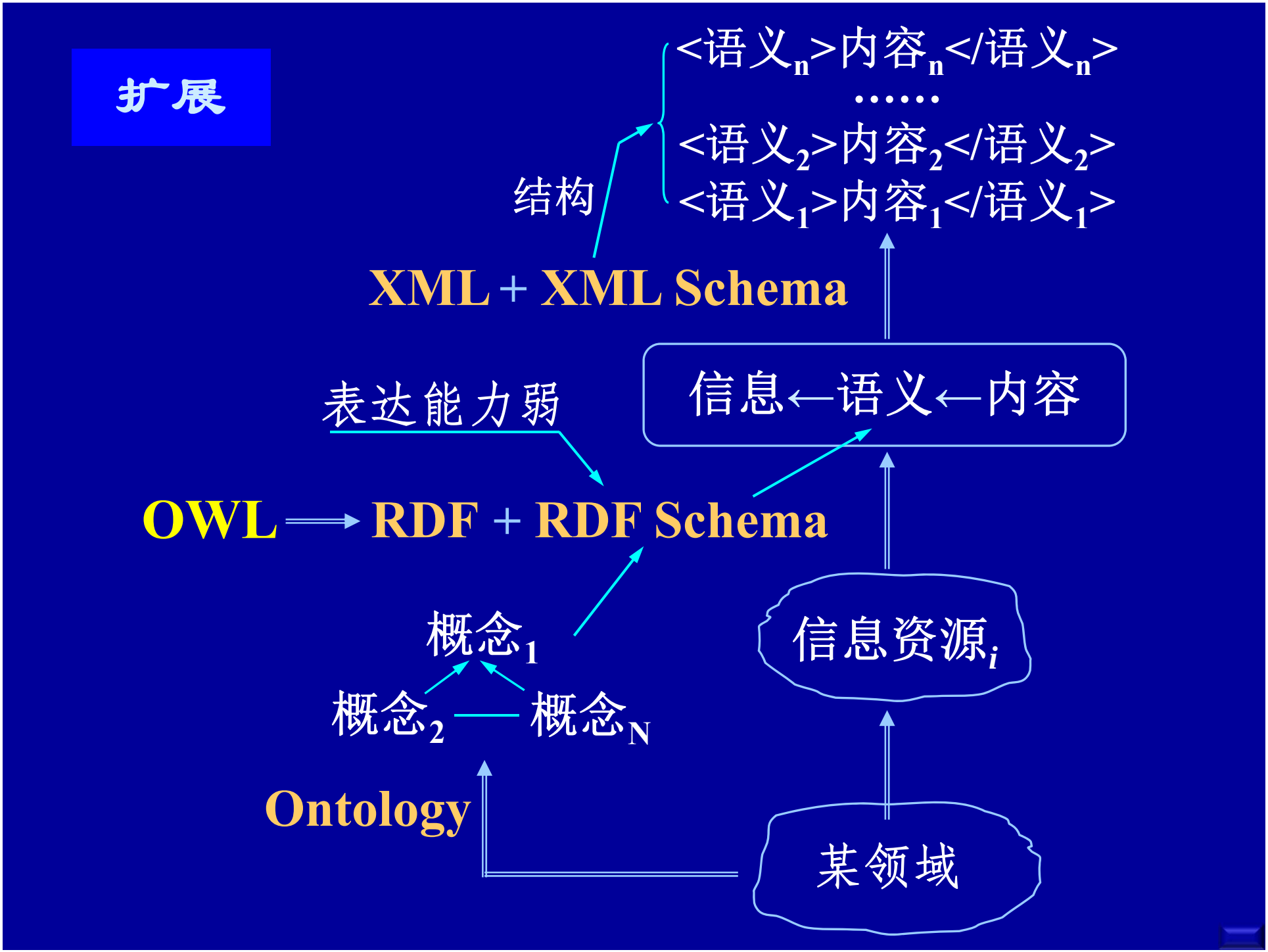

语义网
Trust
规则
Proof
语义
Logic
数字签名
元数据
Ontology
标记文本
RDF
XML
Unicode
URI

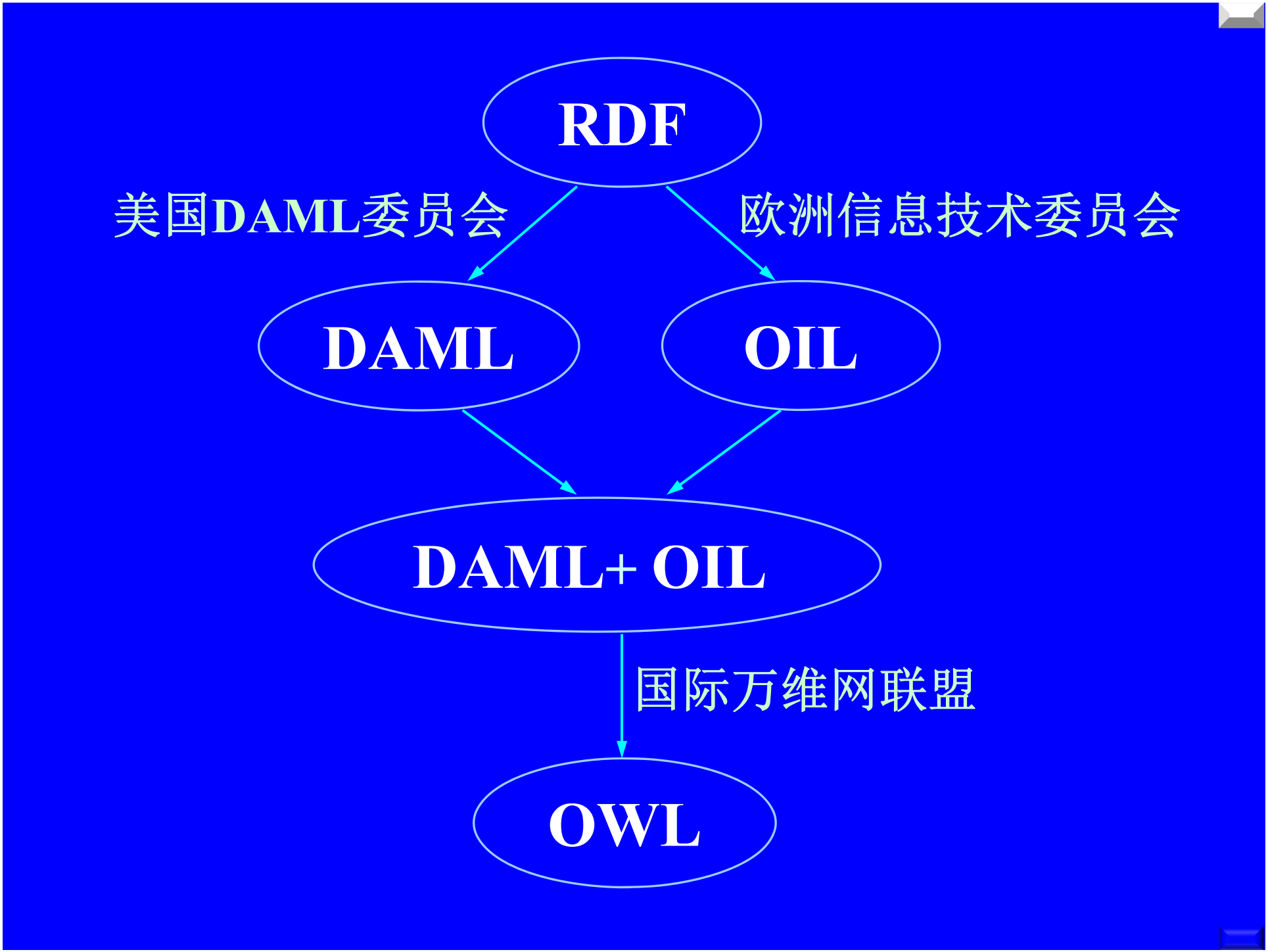

<属性n>属性值n</属性n>
……
<属性2>属性值2</属性2>
<属性1>属性值1</属性1>
结构
XML + XML Schema
资源←属性←属性值
RDF + RDF Schema
信息资源

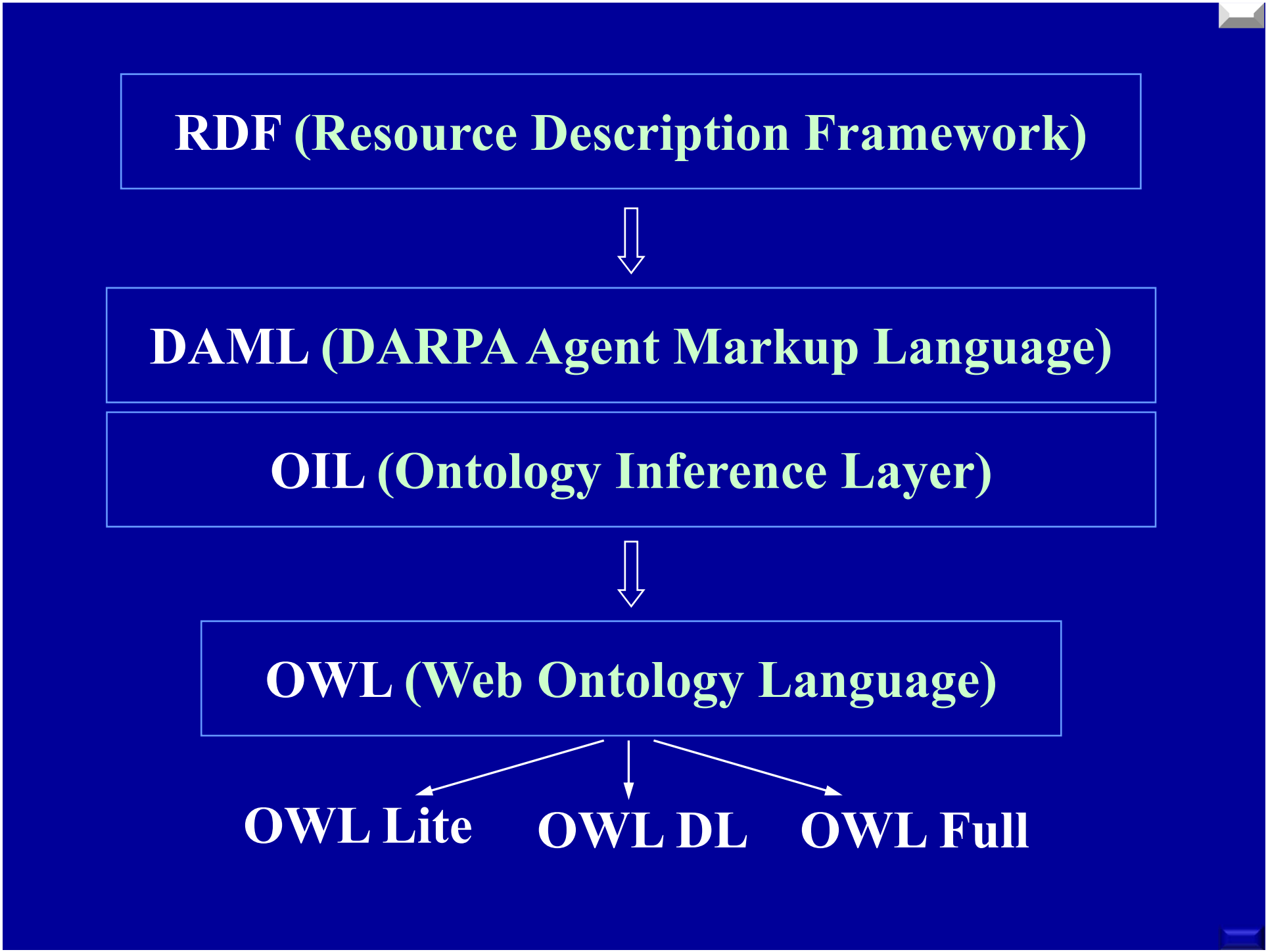

<语义n>内容n</语义n>
.....
<语义2>内容2</语义2>
<语义1>内容1</语义1>
结构
XML + XML Schema
信息←语义←内容
RDF + RDF Schema
信息资源

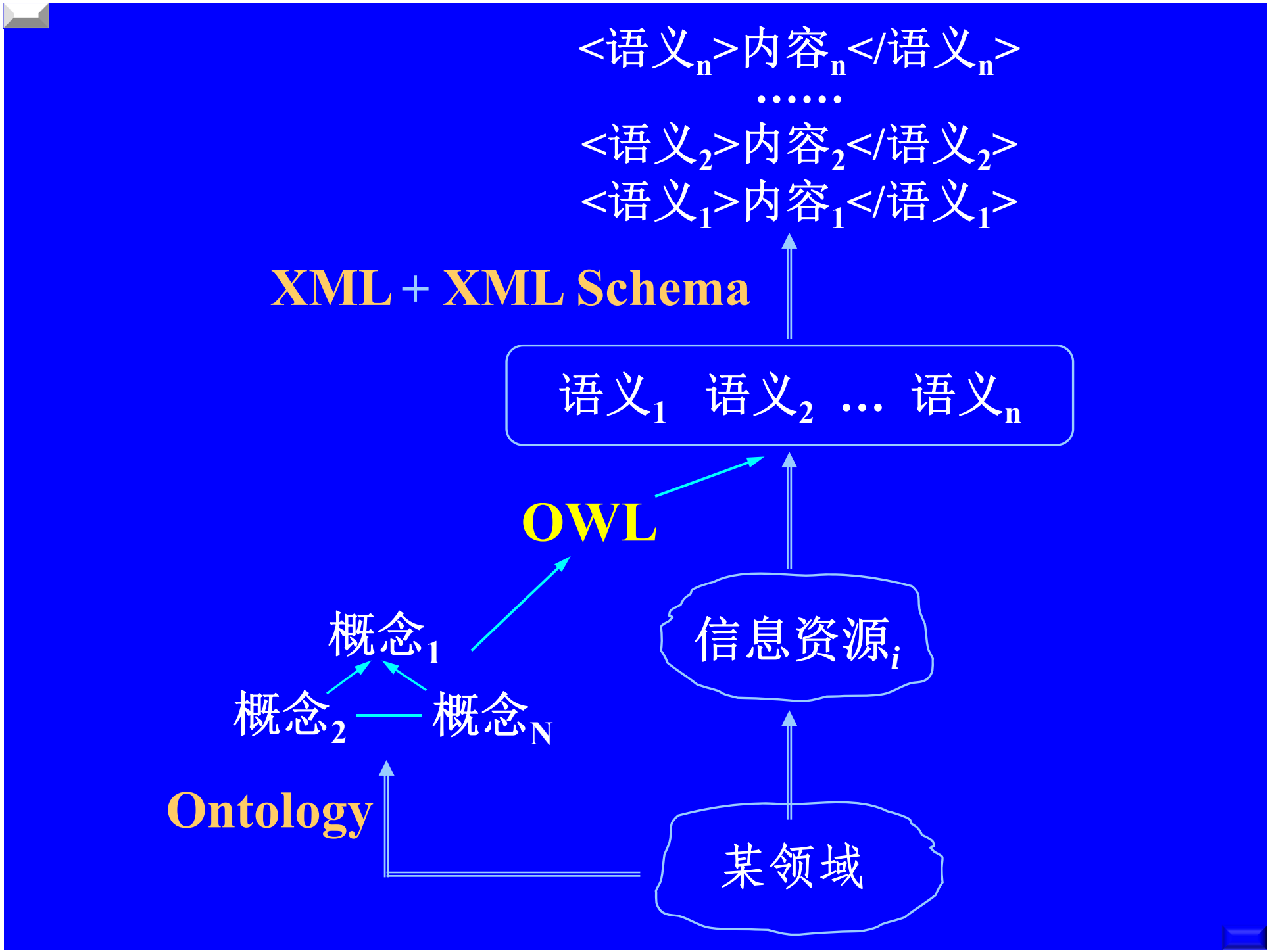

<语义n>内容n</语义n>
……
<语义2>内容2</语义2>
<语义1>内容1</语义1>
结构
XML + XML Schema
信息←语义←内容
RDF + RDF Schema
概念1
概念2
概念N
信息资源
Ontology
领域

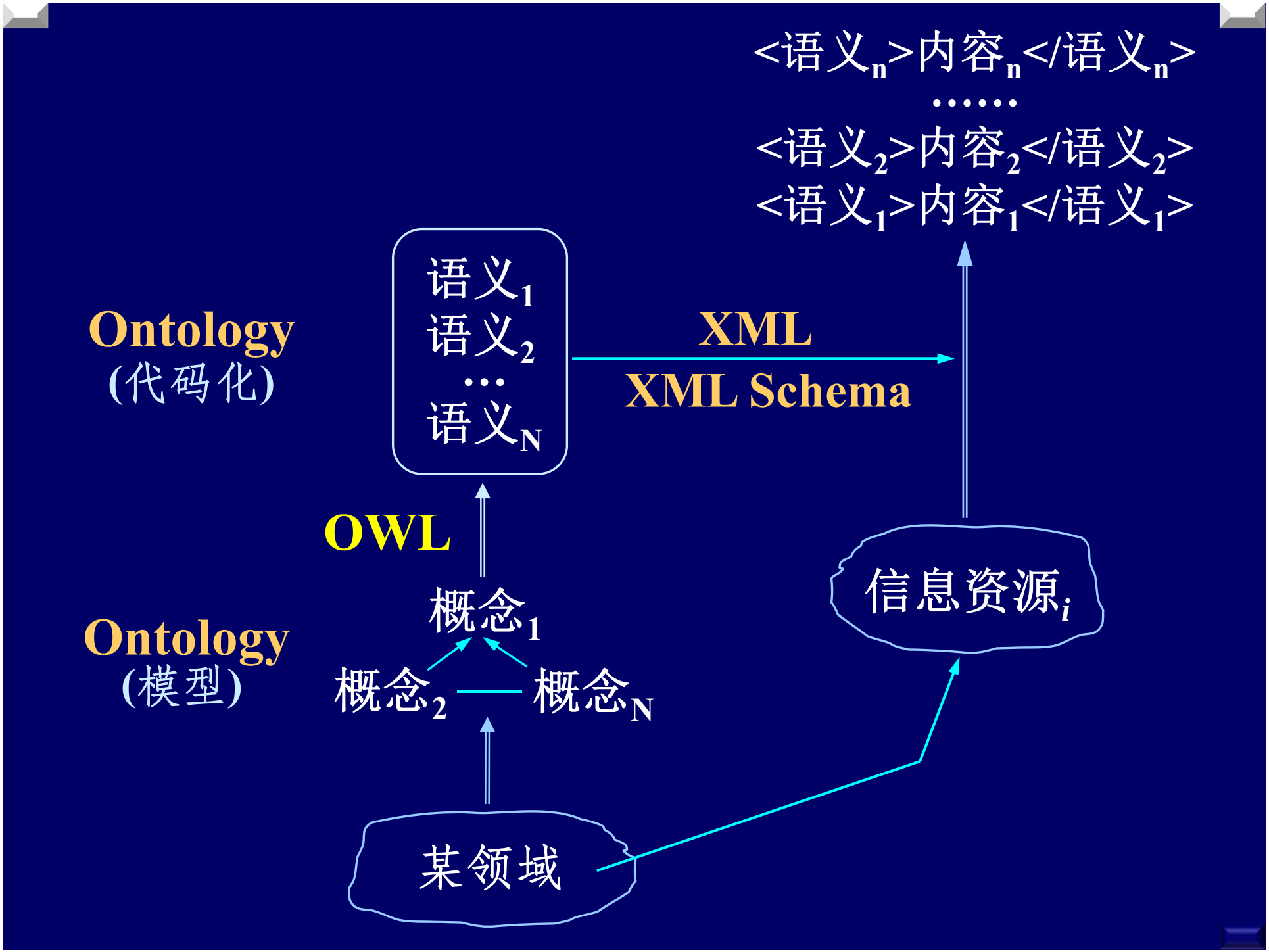

<语义n>内容n</语义n>
……
<语义2>内容2</语义2>
<语义1>内容1</语义1>
结构
XML + XML Schema
信息←语义←内容
RDF + RDF Schema
概念1
概念2
概念N
信息资源i
Ontology
某领域

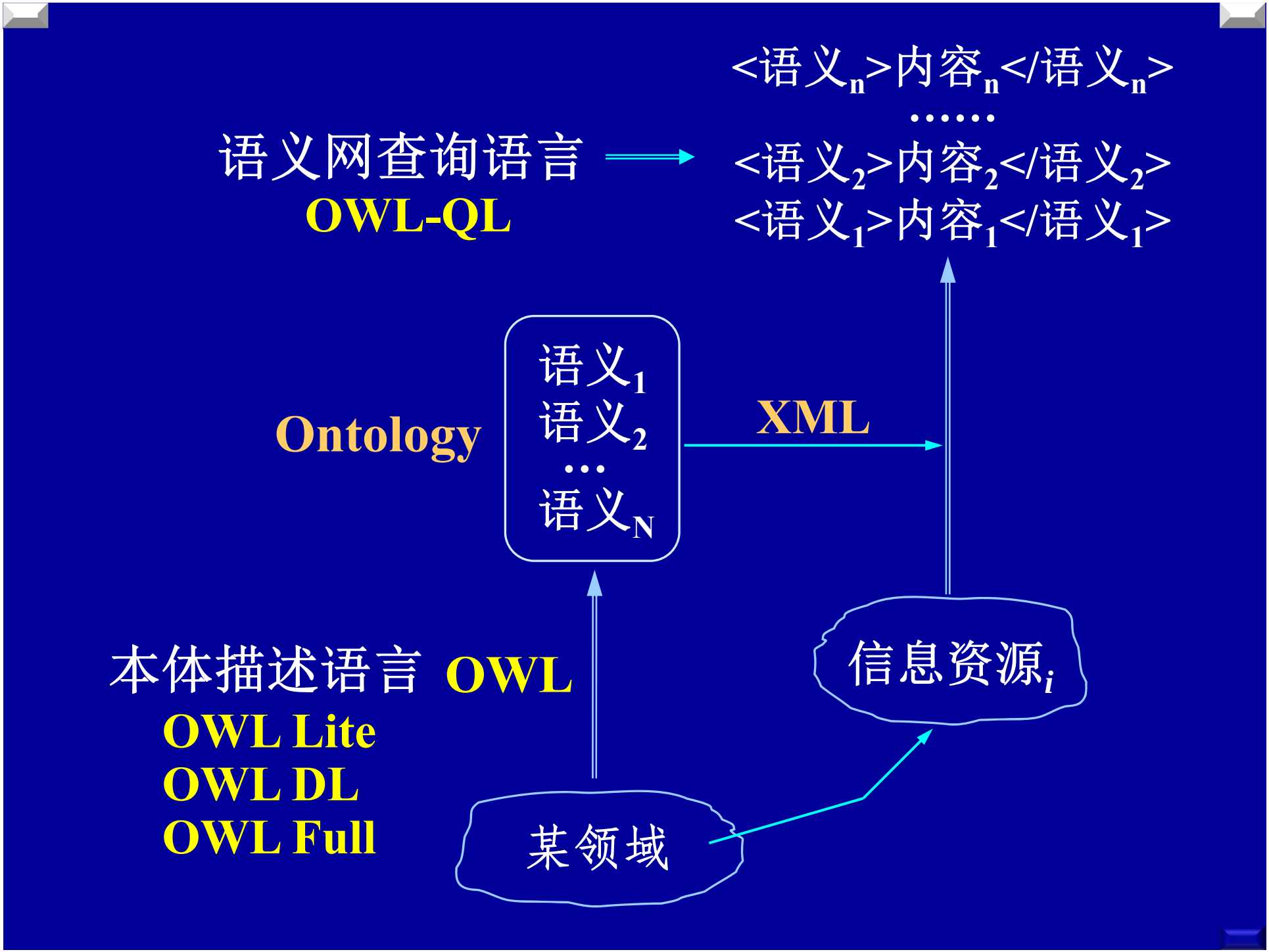

扩展
<语义n>内容n</语义n>
……
<语义2>内容2</语义2>
<语义1>内容1</语义1>
结构
XML + XML Schema
信息←语义←内容
表达能力弱
OWL
RDF + RDF Schema
概念1
概念2
概念N
信息资源i
Ontology
某领域

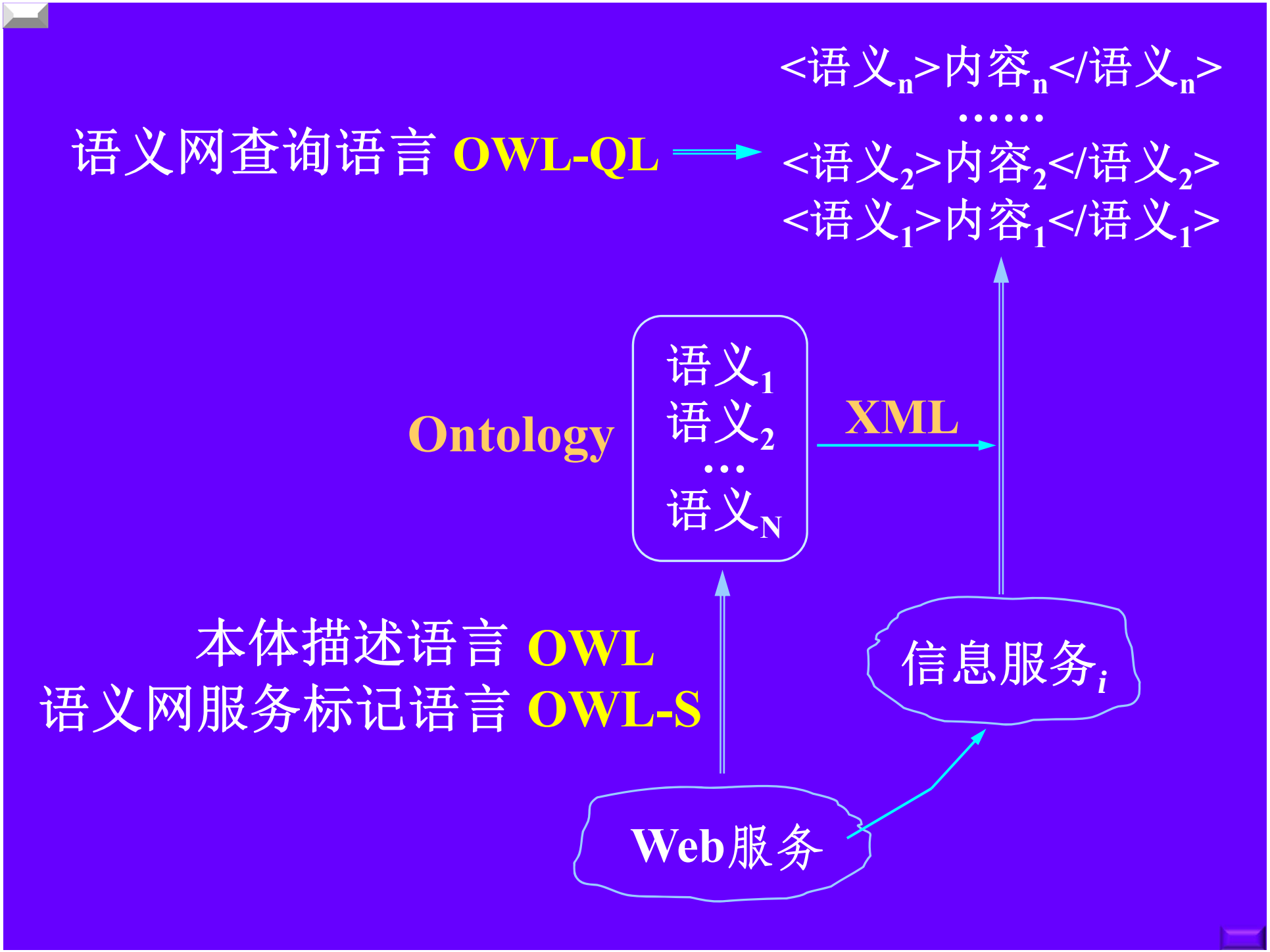

RDF
美国DAML委员会
欧洲信息技术委员会
DAML
OIL
DAML+ OIL
国际万维网联盟
OWL

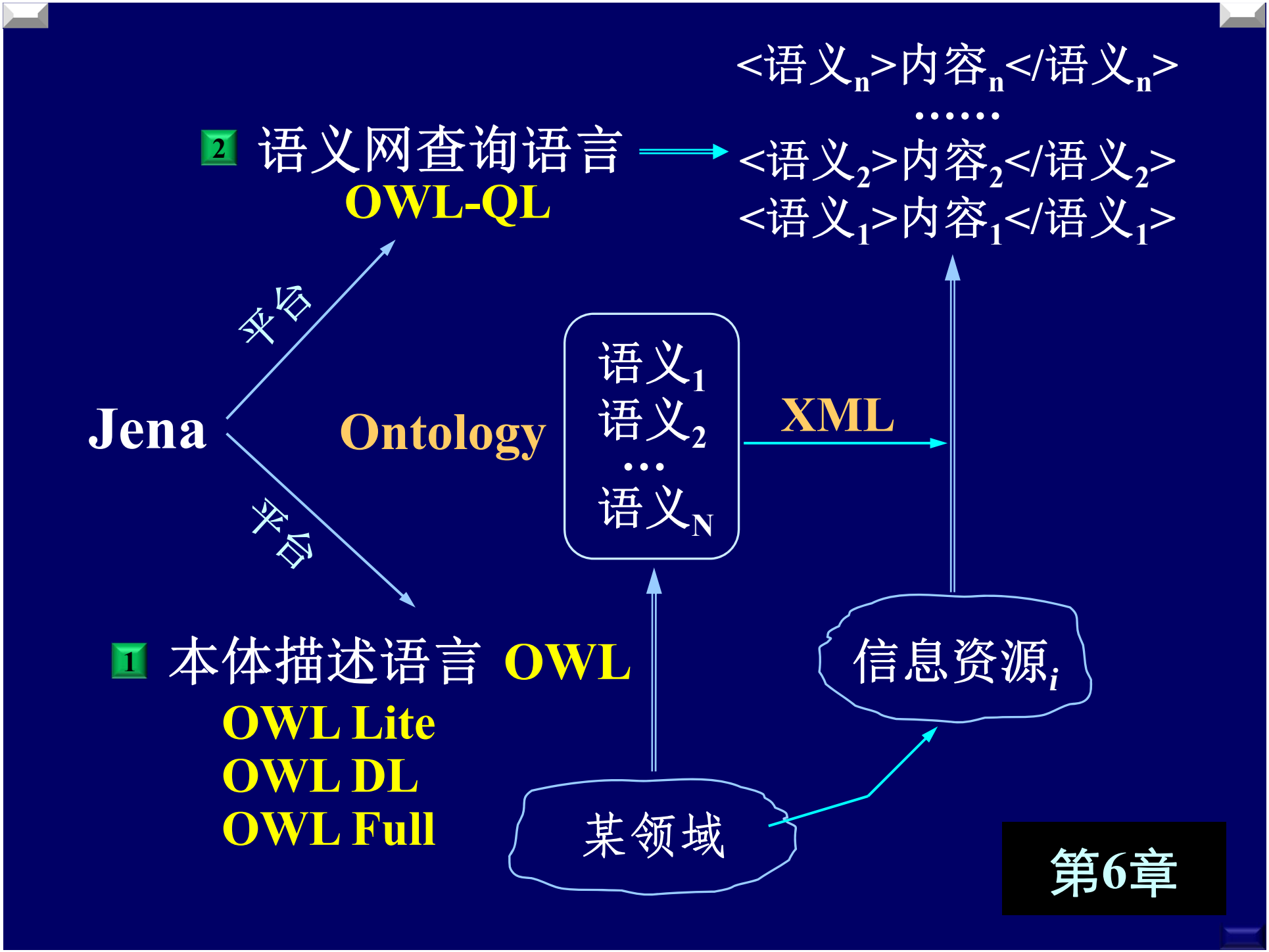

RDF (Resource Description Framework)
DAML (DARPA Agent Markup Language)
OIL (Ontology Inference Layer)
OWL (Web Ontology Language)
OWL Lite
OWL DL
OWL Full

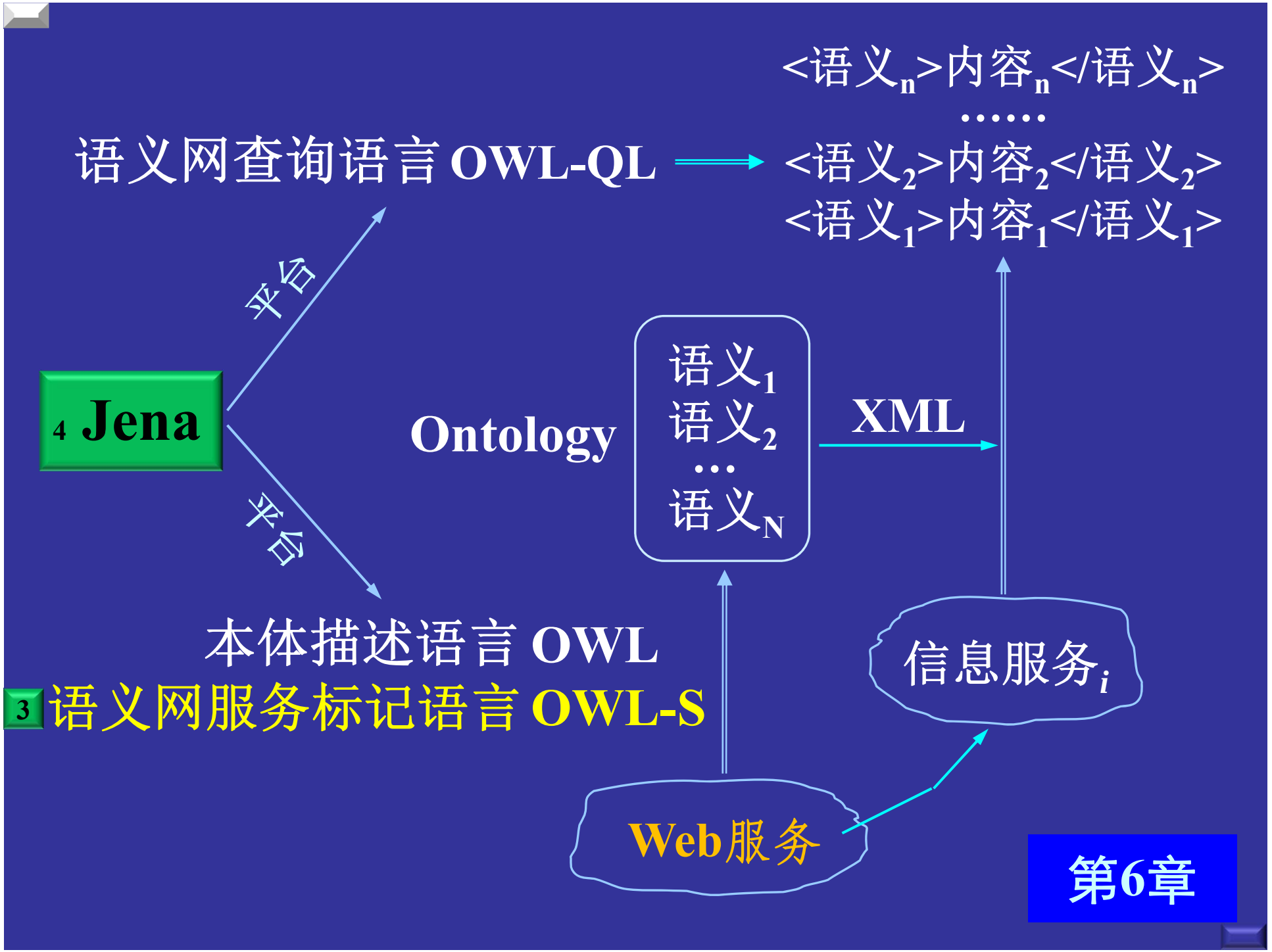

<语义n>内容n</语义n>
……
<语义2>内容2</语义2>
<语义1>内容1</语义1>
XML + XML Schema
语义1 语义2 … 语义n
OWL
概念1
概念2
概念N
Ontology
信息资源i
某领域


<语义n>内容n</语义n>
……
<语义2>内容2</语义2>
<语义1>内容1</语义1>
语义1
语义2
…
语义N
XML
XML Schema
Ontology
(代码化)
OWL
Ontology
(模型)
概念1
概念2
概念N
信息资源i
某领域


<语义n>内容n</语义n>
……
语义网查询语言
OWL-QL
<语义2>内容2</语义2>
<语义1>内容1</语义1>
Ontology
语义1
语义2
…
语义N
XML
本体描述语言 OWL
OWL Lite
OWL DL
OWL Full
某领域
信息资源i


<语义n>内容n</语义n>
……
语义网查询语言 OWL-QL
<语义2>内容2</语义2>
<语义1>内容1</语义1>
Ontology
语义1
语义2
…
语义N
XML
本体描述语言 OWL
语义网服务标记语言 OWL-S
信息服务i
Web服务


<语义n>内容n</语义n>
……
<语义2>内容2</语义2>
<语义1>内容1</语义1>
2 语义网查询语言
OWL-QL
平台
Jena
平台
Ontology
语义1
语义2
…
语义N
XML
1 本体描述语言 OWL
OWL Lite
OWL DL
OWL Full
某领域
信息资源i
第6章


<语义n>内容n</语义n>
……
语义网查询语言 OWL-QL
<语义2>内容2</语义2>
<语义1>内容1</语义1>
平台
4 Jena
平台
Ontology
语义1
语义2
…
语义N
XML
本体描述语言 OWL
3 语义网服务标记语言 OWL-S
信息服务i
Web服务
第6章

Web Ontology Language

网络　本体论　　语言

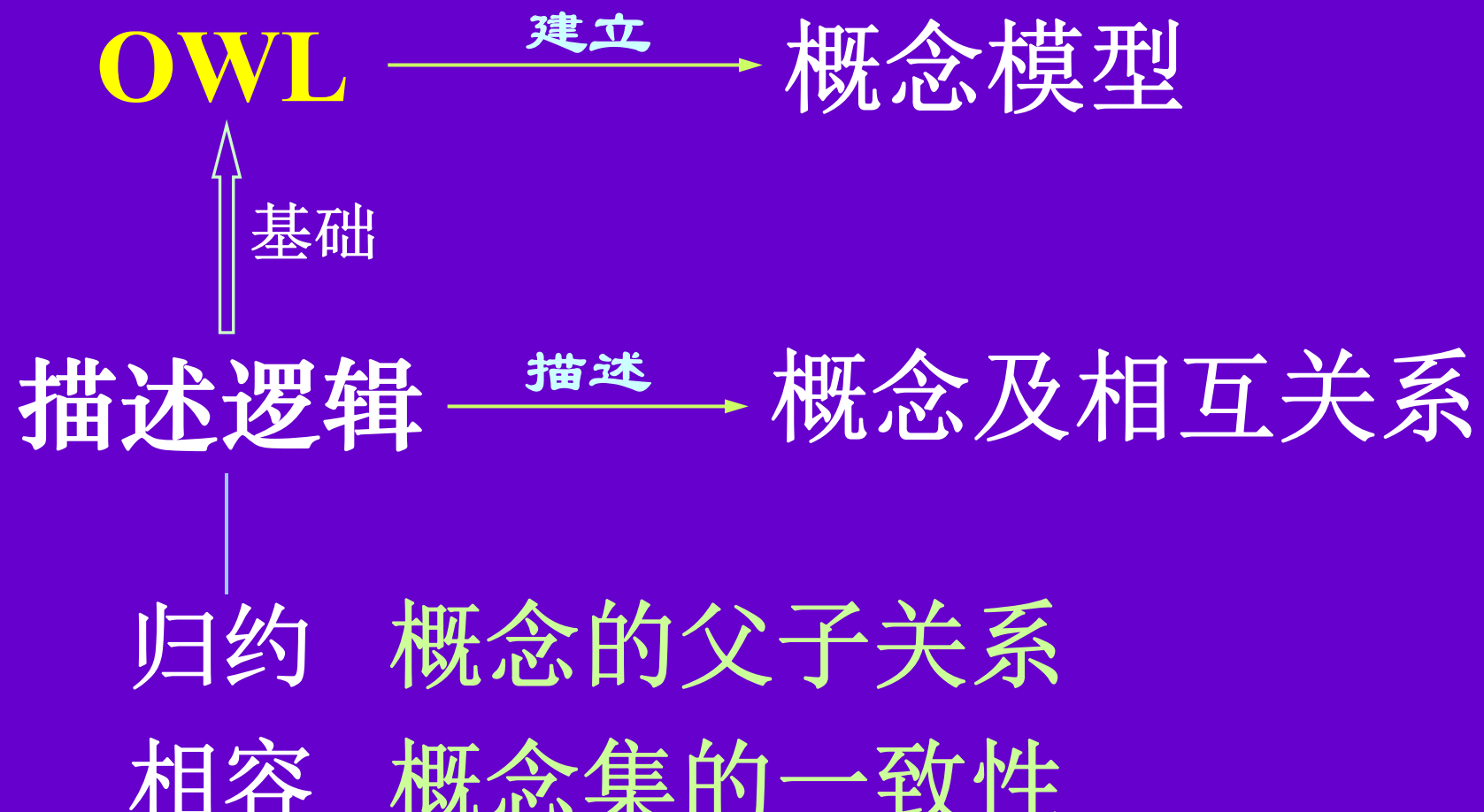
存在　概念模型

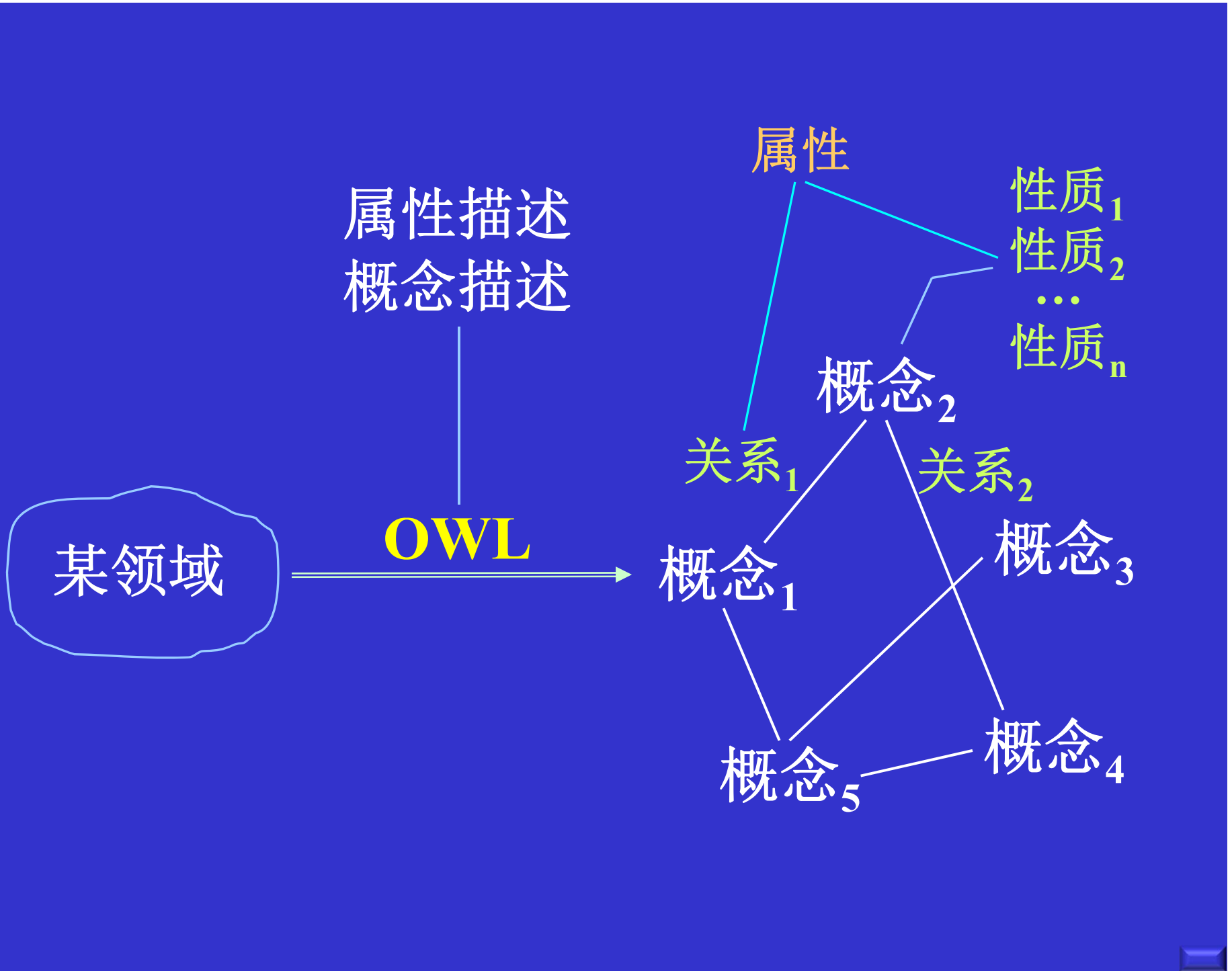
⇩

网络　信息资源　概念模型　建模语言

网络　信息资源　语义模型　建模语言

<语义> 内容 </语义>


OWL
建立
概念模型
基础
描述逻辑
描述
概念及相互关系
归约 概念的父子关系
相容 概念集的一致性


属性描述
概念描述
某领域
OWL
属性
性质1
性质2
…
性质n
概念2
关系1
关系2
概念1
概念3
概念5
概念4

OWL的概念描述

< owl: Class >
表达式$_1$
表达式$_2$
…
表达式$_n$
</owl: Class>

OWL的属性描述

< owl: Property >
表达式$_1$
表达式$_2$
…
表达式$_n$
</owl: Property>

表达式

<构造子> 内容 </构造子>
<构造子 属性=“值”> 内容 </构造子>
<构造子 属性=“值”> </构造子>
<构造子 属性=“值” />

OWL语言结构

< owl: 构造子 >

< 构造子 属性 = “值” / >

< 构造子 属性 = “值” / >

……

< 构造子 属性 = “值” / >

< /owl: 构造子 >

基本构造子(OWL Lite/OWL DL/OWL Full)

RDFS 特征

- Class
  Thing
  Nothing
- rdfs: subClassOf
- rdf: Property
- rdfs: subProperty
- rdfs: domain
- rdfa: range
- Individual

等价性

- equivalentClass
- equivalentProperty
- sameAs
- differentFrom
- AllDifferent
  distinctMembers

类相交

intersectionOf

Class：一个拥有共同性质的个体集合

< owl: Class rdf: ID = “Winery”/ >

< owl: Class rdf: ID = “Region”/ >

< owl: Class rdf: ID = “ConsumableThing”/ >

rdfs: subClassOf: 类的层次结构

```
< owl: Class rdf: ID = “Wine” >
    < rdfs: subClassOf rdf: resource =
                                “#ConsumableThing” / >
                ……
< /owl: Class >
```

rdf: Property：对象属性和数据类型属性

owl: ObjectProperty　owl: DatatypeProperty

```
<owl: Class rdf: ID =“VintageYear”/>
<owl: DatatypeProperty rdf: ID = “yearValue”>
    <rdfs: domain rdf: resource = “#VintageYear”/>
    <rdfs: range rdf: resource = “&xsd; positiveInteger”/>
</owl: DatatypeProperty>
```

rdfs: subProperty：子属性表示属性的层次

<owl: ObjectProperty rdf: ID = “hasSibling”>

<rdfs: subProperty rdf: resource = “#hasRelative”/>

……

</owl: ObjectProperty>

领域 rdfs: domain / 值域 rdfs: range

```
< owl: ObjectProperty rdf: ID = “madeFromGrape” >
    <rdfs: domain rdf: resource = “#Wine”/ >
    <rdfs: range rdf: resource = “#WineGrape”/ >
< /owl:ObjectProperty >
```

Individual: 个体是类的一个实例

< Region rdf: ID = “CentralCoastRegion” / >

owl: equivalentClass　　两个类等价

```
<owl: Class rdf: ID = “Car” >
   <owl: equivalentClass rdf: resource = “Automobile”/>
</owl: Class>
```

owl: equivalentProperty　　两个属性等价

```
<owl: Property rdf: ID = “hasLeader” >
   <owl: equivalentProperty rdf: resource = “hasHead”/>
</owl: Property>
```

owl: sameAs 两个个体相同

```
< Region rdf: ID = “U. S. A” >
   < owl: sameAs rdf: resource=“#theUnitedStates”/ >
< /Region >
```

owl: differentFrom 两个个体不相同

```
< WineSugar rdf: ID = “Dry”/ >
< WineSugar rdf: ID = “Sweet” >
   < owl: differentFrom rdf: resource = “#Dry”/ >
< /WineSuger>
```

owl: AllDifferent：多个个体两两之间互不相同

```
<owl: AllDifferent>
    <owl: distinctMembers rdf: parseType = “Collection”>
        <vin: WineColor rdf: about = “#Red”/>
        <vin: WineColor rdf: about = “#White”/>
        <vin: WineColor rdf: about = “#Rose”/>
    </owl: distinctMembers>
</owl: AllDifferent>
```

基本构造子(OWL Lite/OWL DL/OWL Full)

属性特征

- ObjectProperty
- DatatypeProperty
- inverseOf
- TransitiveProperty
- SymmetricProperty
- FunctionalProperty
- InverseFunctionalProperty

属性约束

- Restriction
- onProperty
- allValuesFrom
- someValuesFrom

基数约束

- minCardinality
- maxCardinality
- cardinality

头信息

- Ontology
- imports

owl: ObjectProperty : 对象属性

```
<owl: ObjectProperty rdf: ID = “hasSibling” >
    <rdfs: subProperty rdf: resource = “#hasRelative”/>
                    ……
</owl: ObjectProperty>
```

owl: DatatypeProperty: 数据类型属性

```
<owl: Class rdf: ID = “VintageYear”>
  <owl: DatatypeProperty rdf: ID = “yearValue”>
    <rdfs: domain rdf: resource = “#VintageYear”/>
    <rdfs: range rdf: resource = “&xsd; positiveInteger”/>
  </owl: DatatypeProperty>
</owl: Class>
```

owl: inverseOf：两个属性互逆

```
< owl: ObjectProperty rdf: ID = “hasMaker” >
   < rdf: type rdf: resource = “&owl; FunctionalProperty”/ >
< /owl: ObjectProperty >

< owl: ObjectProperty rdf: ID = “producesWine” >
    < owl: inverseOf rdf: resource = “#hasMaker”/ >
< /owl: ObjectProperty >
```

owl: TransitiveProperty：属性的传递性

```
<owl: ObjectProperty rdf: ID = “locatedIn”>
   <rdf: type rdf: resource = “&owl; TransitiveProperty”/>
   <rdfs: domain rdf: resource = “&owl; Thing”/>
   <rdfs: range rdf: resource = “#Region”/>
</owl: ObjectProperty>

<Region rdf: ID = “SantaCruzMountainsRegion”>
   <locatedIn rdf: resource = “#CaliforniaRegion”/>
</Region>

<Region rdf: ID = “CaliforniaRegion”>
   <locatedIn rdf: resource = “#USRegion”/>
</Region>
```

owl: SymmetricProperty: 属性的对称性

```
<owl: ObjectProperty rdf: ID = “adjacentRegion” >
   <rdf: type rdf: resource = “&owl; SymmetricProperty”/>
   <rdfs: domain rdf: resource = “#Region”/>
   <rdfs: range rdf: resource = “#Region”/>
</owl: ObjectProperty>

<Region rdf: ID = “MendocinoRegion” >
   <locatedIn rdf: resource = “#CaliforniaRegion”/>
   <adjacentRegion rdf: resource = “#SonomaRegion”/>
</Region>
```

owl: FunctionalProperty : 属性的函数特性

```
<owl: Class rdf: ID = “VintageYear”/>

<owl: ObjectProperty rdf: ID = “hasVintageYear”>
   <rdf: type rdf: resource = “&owl; FunctionalProperty”/>
   <rdfs: domain rdf: resource = “#Vintage”/>
   <rdfs: range rdf: resource = “#VintageYear”/>
</owl: ObjectProperty>
```

owl: InverseFunctionalProperty：属性的反函数特性

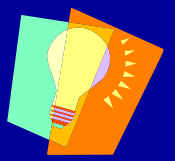
```
<owl: Class rdf: ID = “haxMaker”/>

<owl: ObjectProperty rdf: ID = “producesWine” >
  <rdf: type rdf: resource = “&owl; InverseFunctionalProperty”/>
  <owl: inverseOf rdf: resource = “#hasMaker”/>
</owl: ObjectProperty>
```

owl: allValuesFrom : 限定属性值的来源

```
<owl: Class rdf: ID = “Wine”>
  <rdfs: subClassOf rdf: resource = “&food; PotableLiquid”/>
        ……
  <rdfs: subClassOf>
      <owl: Restriction>
        <owl: onProperty rdf: resource = “#hasMaker”/>
        <owl: allValuesFrom rdf: resource = “#Winery”/>
      </owl: Restriction>
  </rdfs: subClassOf>
        ……
</owl: Class>
```

owl: someValuesFrom : 限定属性值的来源

```
<owl: Class rdf: ID = “Wine”/>
   <rdfs: subClassOf rdf: resource = “&food; PotableLiquid”/>
         ……
   <rdfs: subClassOf>
      <owl: Restriction>
         <owl: onProperty rdf: resource = “#hasMaker”/>
         <owl: someValuesFrom rdf: resource = “#Winery”/>
      </owl: Restriction>
   </rdfs: subClassOf>
         ……
</owl: Class>
```



owl: minCardinality : 限定属性的基数

```
<owl: Class rdf: ID =“Wine”>
  <rdfs: subClassOf rdf: resource =“&food; PotableLiquid”/>
  <rdfs: subClassOf>
    <owl: Restriction>
      <owl: onProperty rdf: resource =“#madeFromGrape”/>
      <owl: minCardinality rdf: datatype=“&xsd; nonNegativeInterge”>
        1
      </owl: minCardinality>
    </owl: Restriction>
  </rdfs: subClassOf>
        ……
</owl: Class>
```

基本构造子(OWL Lite/OWL DL/OWL Full)

版本信息

- versionInfo
- priorVersion
- backwardCompatibleWith
- incompatibleWith
- DeprecatedClass
- DeprecatedProperty

注释属性

- rdfs: label
- rdfs: comment
- rdfs: seeAlso
- rdfs: isDefinedby
- AnnotationProperty
- OntologyProperty

扩展构造子 (OWL DL / OWL Full)

类公理

- one of
- dataRange
- disjointWith
- equivalentClass
- rdfs: subClassOf

类组合

- unionOf
- complementOf
- intersectionOf

基数约束

- minCardinality
- maxCardinality
- cardinality

属性赋值

- hasValue

owl: oneOf : 枚举个体定义类

```
< owl: Class rdf: ID = “WineColor” >
   < rdfs: subClassOf rdf: resource = “#WineDescriptor”/ >
   < owl: oneOf rdf: parseType = “Collection” >
      < owl: Thing rdf: about = “#White”/ >
      < owl: Thing rdf: about = “#Rose”/ >
      < owl: Thing rdf: about = “#Red”/ >
   < /owl: oneOf >
</owl: Class>
```

owl: disjointWith: 类与类之间互斥

```
<owl: Class rdf: ID = “Pasta”>
    <rdfs: subClassOf rdf: resource = “#EdibleThing”/>
    <owl: disjointWith rdf: resource = “#Meat”/>
    <owl: disjointWith rdf: resource = “#Fowl”/>
    <owl: disjointWith rdf: resource = “#Seafood”/>
    <owl: disjointWith rdf: resource = “#Dessert”/>
    <owl: disjointWith rdf: resource = “#Fruit”/>
</owl: Class>
```

owl: unionOf 类或属性的并集

```
<owl: Class rdf: ID = “Fruit”>
   <owl: unionOf rdf: parseType = “Collection”>
      <owl: Class rdf: about = “#SweetFruit”/>
      <owl: Class rdf: about = “#NonSweetFruit”/>
   </owl: unionOf>
</owl: Class>
```

owl: hasValue : 指定属性的至少一个值

```
< owl: Class rdf: ID = “Burgundy” >
    ……
   < rdfs: subClassOf >
     < owl: Restriction >
       < owl: onProperty rdf: resource = “#hasSugar”/ >
       < owl: hasValue rdf: resource = “#Dry”/ >
     < /owl: Restriction/ >
   < /rdfs: subClassOf >
    ……
< /owl: Class >
```

OWL-QL的特点 (P152)

- 不依赖具体形式的知识库
- 为用户提供预选的集合
- 语义信息表达语义相互关系
- 高层抽象与具体接口无关

图6-2 OWL-QL查询过程

查询(Query)

必需内容

- 查询模式：OWL-QL语句
- AnswerKB：引用的知识库
- Must-Bind变量列表：所查询变量

可选内容

- 应答模式(Answer Pattern)
- 查询假设(Query Premise)
- 认证请求(Justification Request)
- 应答簇大小(Answer Bundle Size)

应答簇(Answer Bundle)

- 对应查询的一系列Answer
- Process Handle / 终止符号

End
None
Rejected

应答

必有项

- 应答模式实例
- 所响应的查询
- 响应的服务器

可选项

- 应答认证(**Answer Justification**)

服务继续 (Server Continuation)

- **Process Handle**
- 应答簇大小(可选)

服务终止 (Server Termination)

- **Process Handle**

OWL-QL示例 (P155)

If C1 is a Seafood Course
W1 is a drink of C1
What color is W1?

- 查询: (type C1 Seafood-Course)
  (drink C1 W1)
  (has-color W1 ?x)
  Must-bind ?x

  } 查询模式

  变量列表

  应答: White

查询(Query)

必需内容

- 查询模式：OWL-QL语句
- AnswerKB：引用的知识库
- Must-Bind变量列表：所查询变量

可选内容

- 应答模式(Answer Pattern)
- 查询假设(Query Premise)
- 认证请求(Justification Request)
- 应答簇大小(Answer Bundle Size)

OWL-QL应用示例

If C1 is a Seafood Course
W1 is a drink of C1
What color is W1?

查询消息

应答消息

```
<owl-ql: query
    xmlns: owl-ql = “http://www.w3.org/2003/10/owl-ql-syntax#”
    xmlns: var = “http://www.w3.org/2003/10/owl-ql-variables#”>
        <owl-ql: premise>
            …… 1
        </owl-ql: premise>
        <owl-ql: queryPattern>
            …… 2
        </owl-ql: queryPattern>
        <owl-ql: mustBindVars>
            …… 3
        </owl-ql: mustBindVars>
        <owl-ql: answerKBPattern>
            …… 4
        </owl-ql: answerKBPattern>
        <owl-ql: answerSizeBound>
            …… 5
        </owl-ql: answerSizeBound>
</owl-ql: query>
```

查询消息

查询条件

```
<owl-ql: premise>
    <rdfs: RDF>
        <rdf: Description rdf: about = “#C1”>
            <rdf: type rdf: resource = “#Seafood-Course”/>
            <drink rdf: resource = “#W1”/>
        </rdf: Description>
    </rdfs: RDF>
</owl-ql: premise>
```

OWL-QL应用示例

If C1 is a Seafood Course
W1 is a drink of C1
What color is W1?

查询模式

```
<owl-ql: queryPattern>
  <rdfs: RDF>
    <rdf: Description rdf: about = “#W1”>
      <has-color rdf: resource =
        “http//www.w3.org/2003/10/owl-ql-variables#x”/>
    </rdf: Description>
  </rdfs: RDF >
</owl-ql: queryPattern>
```

```
<owl-ql: mustBindVars>
    <var: x/>
</owl-ql: mustBindVars>

<owl-ql: answerKBPattern>
    <owl-ql: kbRef rdf: resource =
          “http//ontolingua.stanford.edu/wines.owl”/>
</owl-ql: answerKBPattern>

<owl-ql: answerSizeBound>
     5
</owl-ql: answerSizeBound>
```

变量列表

知识库

应答簇

```
<owl-ql: answerBundle
    xmlns: owl-ql = “http://www.w3.org/2003/10/owl-ql-syntax#”
    xmlns: var = “http://www.w3.org/2003/10/owl-ql-variables#”>
        <owl-ql: queryPattern>
            …… 1
        </owl-ql: queryPattern>
        <owl-ql: answer>
            …… 2
        </owl-ql: answer>
        <owl-ql: continuation>
            …… 3
        </owl-ql: continuation>
</owl-ql: answerBundle>
```

应答消息

应答簇(Answer Bundle)

- 对应查询的一系列Answer
- Process Handle / 终止符号

End
None
Rejected

应答

必有项

- 应答模式实例
- 所响应的查询
- 响应的服务器

可选项

- 应答认证(**Answer Justification**)

查询模式

```
<owl-ql: queryPattern>
  <rdfs: RDF>
     <rdf: Description rdf: about = “#W1”>
        <has-color rdf: resource =
           “http//www.w3.org/2003/10/owl-ql-variables#x”/>
     </rdf: Description>
  </rdfs: RDF >
</owl-ql: queryPattern>
```

应答

```
<owl-ql: answer>
    <owl-ql: binding-set>
        <var: x rdf: resource = “#White”/>
    </owl-ql: binding-set>
    <owl-ql: answerPatternInstance>
        <rdfs: RDF>
            <rdf: Description rdf: about = “#W1”>
                <has-color rdf: resource = “#White”/>
            </rdf: Description>
        </rdfs: RDF >
    </owl-ql: answerPatternInstance>
</owl-ql: answer>
```

终止符号(Server Termination)

```
< owl-ql: continuation >
    < owl-ql: termination-token >
        < owl-ql: none/ >
    < /owl-ql: termination-token >
< /owl-ql: continuation >
```

语义网服务标记语言OWL-S

用本体语言对网页中的服务进行语义标记，使服务变成机器可理解、Agent可识别的，促使网络服务的互操作向自动化方向发展。

Web服务 —用本体论进行语义标记（2）→ 语义网服务（1）

OWL-S的四种目标 (P158)

(1) 自动Web服务发现
(2) 自动Web服务触发
(3) 自动Web服务组合和互操作
(4) 自动Web服务执行监控

语义网服务的实现途径(P162)

(1) 构建一个Agent，搜集Web服务实体的各种元数据信息；

(2) 基于Web服务本体对这些服务进行分析整合，创建一个分布式知识库；

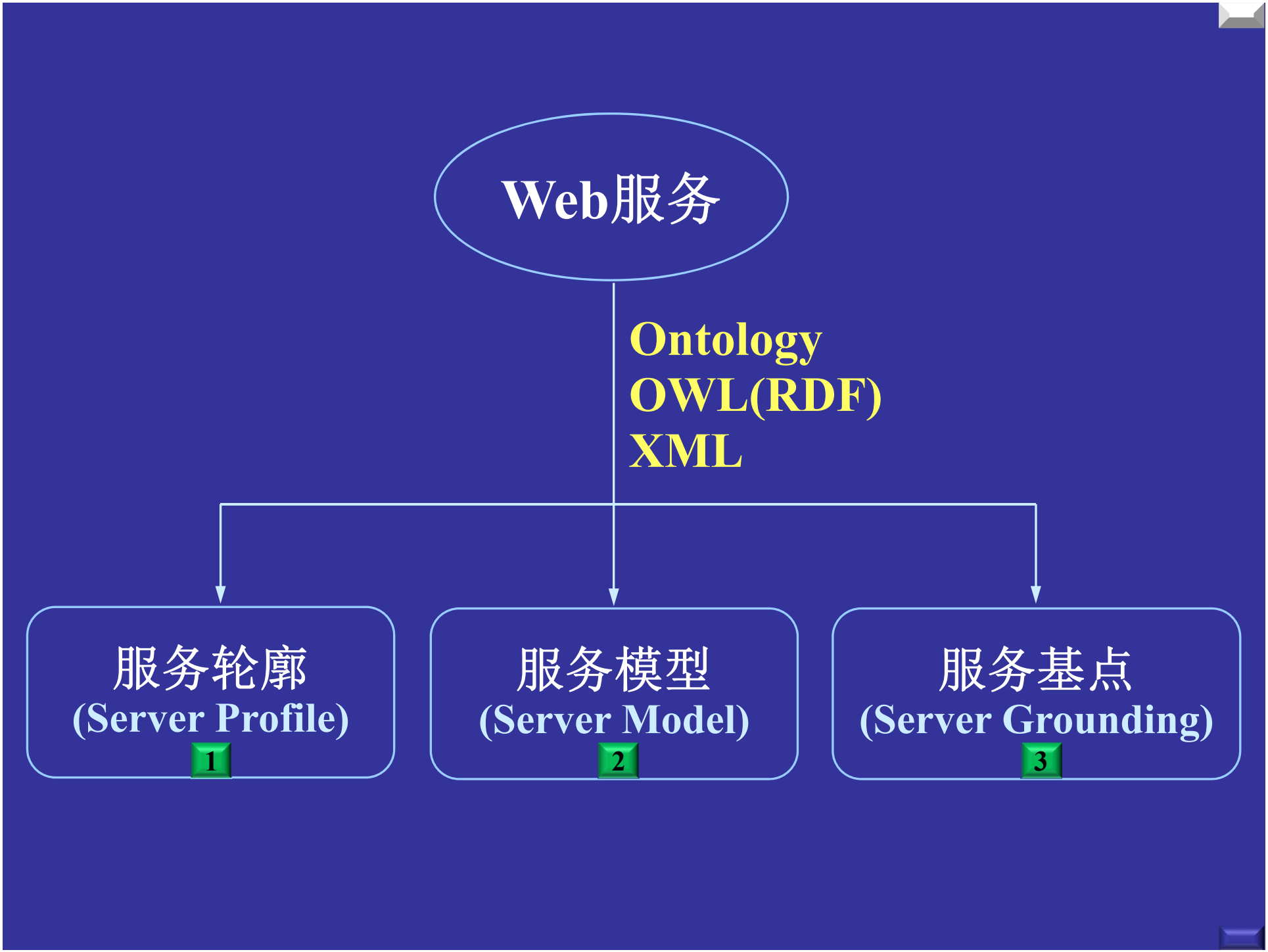
(3) 将这个知识库提供给Agent，使它可以通过自动推理自动发现服务、执行服务、组合服务和互操作。

基于本体描述

Service Profile 是提供什么的？

Service Model 是如何工作的？

Service Grounding 是如何被访问的？


Web服务
Ontology
OWL(RDF)
XML
服务轮廓
(Server Profile)
1
服务模型
(Server Model)
2
服务基点
(Server Grounding)
3

服务轮廓的标记方法

<profile: profile rdf: ID =“Profile My_Web_Service”>

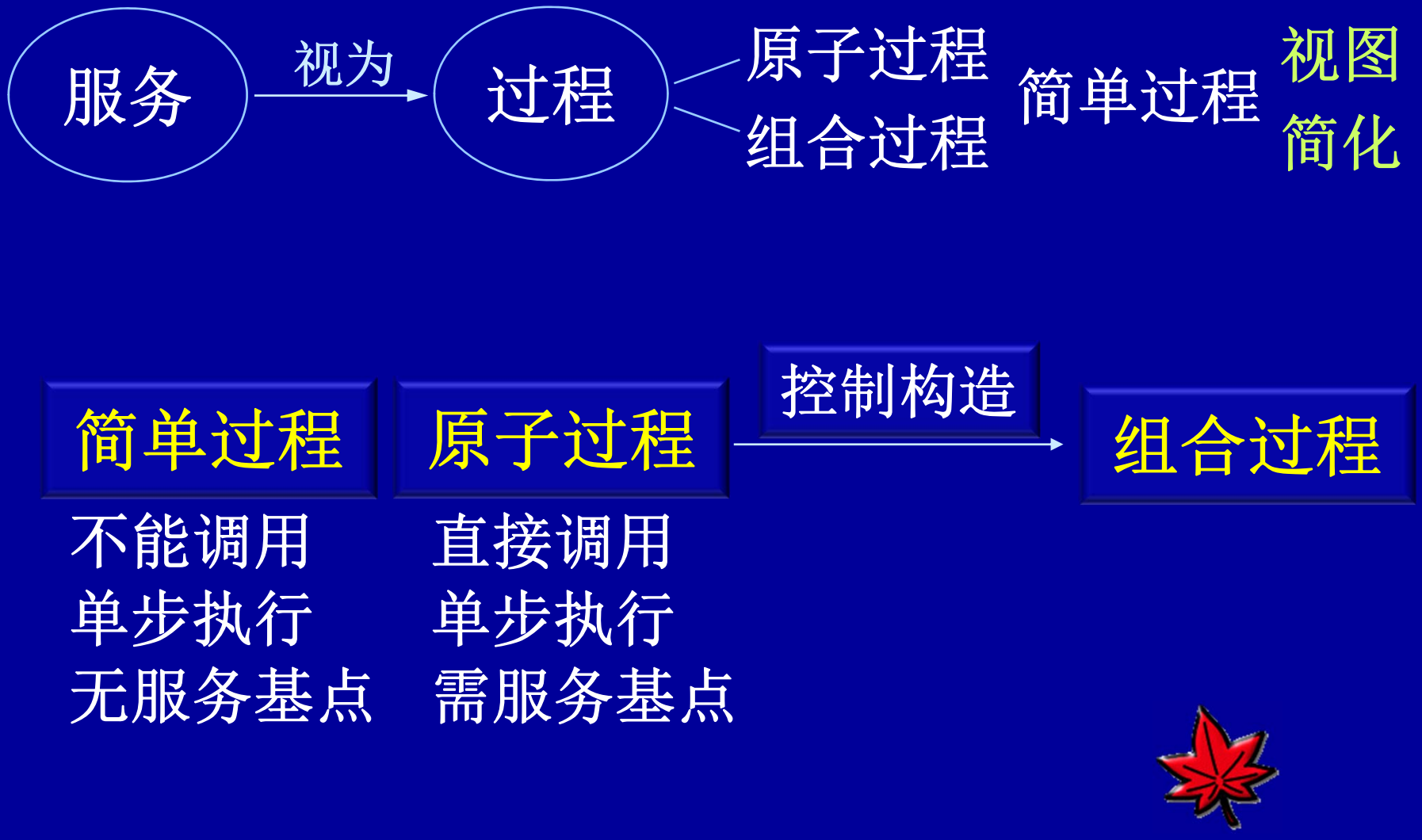
服务轮廓的描述属性

presents
presentsBy

Input
Output
Precondition
Effect

serviceName
textDescription
contactInformation

serviceParameter
serviceCategory
QualityRating


服务
视为
过程
原子过程
组合过程
简单过程
视图
简化
简单过程
原子过程
控制构造
组合过程
不能调用
单步执行
无服务基点
直接调用
单步执行
需服务基点

过程的类型及其定义(P160)

```
<owl: Class rdf: ID = “process” >
    <rdf: comment>
        The most general class of processes
    </rdf: comment>
    <owl: disjointUnionOf rdf: parseType = “owl collection”>
        <owl: Class rdf: about = “#AtomicProcess”/>
        <owl: Class rdf: about = “#SimpleProcess”/>
        <owl: Class rdf: about = “#CompositeProcess”/>
    </owl: disjointUnionOf>
</owl: Class>
```

原子过程 (Atomic Process)

```
<owl: Class rdf: ID = “AtomicProcess” >
    <owl: subClassOf rdf: resource = “#Process”/>
</owl: Class>
```

简单过程的类及其性质(P161)

```
<owl: Class rdf: ID = “SimpleProcess” >
    <owl: subClassOf rdf: resource = “#Process”/>
</owl: Class>
<rdf: Property rdf: ID = “RealizedBy” >
    <rdfs: domain rdf: resource = “#AtomicProcess”/>
    <rdfs: range rdf: resource = “#AtomicProcess”/>
    <owl: inverseOf rdf: resource = “#realizes”/>
</rdf: Property>
<rdf: Property rdf: ID = “ExpandsTo” >
    <rdfs: domain rdf: resource = “#SimpleProcess”/>
    <rdfs: range rdf: resource = “#CompositeProcess”/>
    <owl: inverseOf rdf: resource = “#collapsesTo”/>
</rdf: Property>
```

```
<owl: Class rdf: ID = “CompositeProcess” >
   <owl: intersectionOf rdf: parseType = “owl: collection”>
      <owl: Class rdf: about = “Process”/ >
      <owl: Restriction owl: Cardinality + “1” >
         <owl: onProperty rdf: resource = “#composedOf”/>
      </owl: Restriction>
   </owl: intersectionOf>
</owl: Class>
```

组合过程

控制构造的表达方式(P161)

```
<rdf: Property rdf: ID = “composedOf” >
  <rdfs: domain rdf: resource = “#CompositeProcess”/>
  <rdfs: range rdf: resource = “#ControlConstruct”/>
</rdf: Property>
```

服务基点(Service Grounding)

描述如何获取服务的细节

协议　消息格式　序列化
传输　寻址方式

目前主流的Ontology工具

Protege

(独立应用程序)

Jena

(Eclipse插件)

语义网

〈元数据〉内容〈/元数据〉

〈语义〉内容〈/语义〉

〈属性〉属性值〈/属性〉

RDF

资源 ← 属性 ← 属性值

三元组　｛属性，资源，属性值｝**statement**

predicate subject object

访问RDF | **创建RDF** | **查询RDF**

读取文件中的RDF (P164)

```
File f;
FileReader fr;
Model model;
f = new File("C:\test.html");
fr = new FileReader(f);
model = new ModelMem( );
model.read(fr, RDFS.getURI( ));
```

读取全部Statement (P165)

```
Model model;
StmtIterator iter;
Statement stmt;

iter = model.listStatements( );
while(iter.hasNext( ))
{
    stmt = iter.next( );
}
```

访问RDF三元组

```
Property predicate;
Resource subject;
RDFNode obj;
Statement stmt;
subject = stmt.getSubject( );
System.out.printIn("Subject = "+subject.getURI( ));
predicate = stmt.getPredicate( );
System.out.printIn("Predicate = "+predicate.getLocalName( ));
obj = stmt.getObject( );
System.out.printIn("Object = "+obj.toString( ));
```

Jena创建RDF三元组

```
Model model;
String namespace = “http://www.test.com”;
model.createResource(http://www.foo.com/boats#sailboat)
  .addproperty(model.createProperty(namespace, “length”), 25)
  .addproperty(model.createProperty(namespace, “color”), “teal”);
```

{ x:length, http://www.foo.com/boats#sailboat, 25 }

{ x:color, http://www.foo.com/boats#sailboat, “teal” }

查询与非洲旅行有关的信息

```
Model model;
Resource r;
ResIterator resourceIter;
resourceIter = model.listSubjectsWithProperty
   (model.createPreperty("http://foo.org/destination"), "Africa");
while(resourceIter.hasNext( ))
{
     r = resourceIter.next( );
     System.out.printIn("Resource" + r.toString( ) +
                                          "is about travel to Africal");
}
```

OWL推理

Schema文件

Data文件

⇩

推理器 ⇨

查找实例

认证实例

检查数据

```
< ? xml version = “1.0” ? >
< ! DOCTYPE rdf: RDF [
    <! ENTITY eg ‘urn: x-hp: eg/’ >
    <! ENTITY rdf ‘http://www.w3.org/1999/02/22-rdf-syntax-ns#’>
    <! ENTITY rdfs ‘http://www.w3.org/2000/01/rdf-schema#’>
    <! ENTITY xsd ‘http://www.w3.org/2001/XMLSchama#’>
    <! ENTITY owl ‘http://www.w3.org/2002/07/owl#’>
] >
< rdf: RDF xmlns: rdf = “&rdf;”
           xmlns: rdfs = “&rdfs;”
           xmlns: xsd = “&xsd;”
           xmlns: owl = “&owl;”
           xmlns: base = “urn: x-hp: eg/”
           xmlns: eg = “&eg;” >
        ……
    类定义和属性定义
        ……
</rdf: RDF>
```

owlDemoSchema.xml

```
< owl: Class rdf: about = “&eg; Computer” >
        ……
< /owl: Class >
< owl: Class rdf: about = “&eg; MotherBoard”/ >
< owl: Class rdf: about = “&eg; GraphicsCard”/ >
< owl: Class rdf: about = “&eg; Bundle”/ >
< owl: Class rdf: about = “&eg; GameBundle” >
   < rdfs: subClassOf rdf: resource = “&eg; Bundle” />
< /owl: Class >

< eg: GraphicsCard rdf: about = “&eg; budgetGraphics”/ >
< eg: GraphicsCard rdf: about = “&eg; gamingGraphics”/ >
< eg: GraphicsCard rdf: about = “&eg; DTPGraphics”/ >

< owl: ObjectProperty rdf: about = “&eg; hasComponent” >
   < rdf: type rdf: resource = “&owl; TransitiveProperty” />
< /owl: ObjectProperty >
```

```
<owl: Class rdf: about = “eg; Computer”>
  <rdfs: subClassOf>
    <rdf: Description>
      <owl: intersectionOf rdf: parseType = “Collection”>
        <owl: Restriction>
          <owl: onProperty rdf: resource = “&eg; hasMotherBoard”/>
          <owl: maxCardinality rdf: datatype =
                                        “&xsd; nonNegativeInteger”>
              1
          </owl: maxCardinality>
        </owl: Restriction>
      </owl: intersectionOf>
    </rdf: Description>
  </rdfs: subClassOf>
</owl: Class>
```

```
<owl: ObjectProperty rdf: about = “&eg; hasGraphics”>
  <rdfs: range rdf: resource = “&eg; GraphicsCard”/>
  <rdfs: subPropertyOf rdf: resource = “&eg; hasComponent”/>
</owl: ObjectProperty>

<owl: ObjectProperty rdf: about = “&eg; hasMotherBoard”>
  <rdfs: range rdf: resource = “&eg; MotherBoard”/>
  <rdfs: domain rdf: resource = “&eg; Computer”/>
  <rdfs: subPropertyOf rdf: resource = “&eg; hasComponent”/>
</owl: ObjectProperty>

<owl: ObjectProperty rdf: about = “&eg; hasBundle”>
   <rdfs: domain rdf: resource = “&eg; Computer”/>
</owl: ObjectProperty>

<owl: Class rdf: about = “&eg; GamingComputer”>
        ……
</owl: Class>
```

```
<owl: Class rdf: about = “eg; GamingComputer”>
  <owl: equivalentClass>
    <rdf: Description>
      <owl: intersectionOf rdf: parseType = “Collection”>
        <owl: Restriction>
          <owl: onProperty rdf: resource = “&eg; hasComponent”/>
          <owl: hasValue rdf: resource = “&eg; gamingGraphics”/>
        </owl: Restriction>
        <owl: Restriction>
          <owl: onProperty rdf: resource = “&eg; hasBundle”/>
          <owl: someValueFrom rdf: resource = “&eg; GameBundle”/>
        </owl: Restriction>
        <owl: Class rdf: about = “&eg; Computer”/>
      </owl: intersectionOf>
    </rdf: Description>
  <owl: equivalentClass>
</owl: Class>
```

```
< ? xml version = “1.0” ? >
< ! DOCTYPE rdf: RDF [
    <! ENTITY eg ‘urn: x-hp: eg/’ >
    <! ENTITY rdf ‘http://www.w3.org/1999/02/22-rdf-syntax-ns#’>
    <! ENTITY rdfs ‘http://www.w3.org/2000/01/rdf-schema#’>
    <! ENTITY xsd ‘http://www.w3.org/2001/XMLSchama#’>
    <! ENTITY owl ‘http://www.w3.org/2002/07/owl#’>
] >
< rdf: RDF xmlns: rdf = “&rdf;”
           xmlns: rdfs = “&rdfs;”
           xmlns: xsd = “&xsd;”
           xmlns: owl = “&owl;”
           xmlns: base = “urn: x-hp: eg/”
           xmlns: eg = “&eg;” >
        ……
    类的实例描述
        ……
</rdf: RDF>
```

owlDemoData.xml

```
<Computer rdf: about = “&eg; whiteBoxZX”>
    <hasMotherBoard rdf: resource = “&eg; nForce”/>
    <hasbundle>
        <GameBundle rdf: about = “&eg; actionPack”/>
    </hasbundle>
</Computer>
<Computer rdf: about = “&eg; whiteBoxZX”>
    <hasMotherBoard>
        <rdf: Description rdf: about = “&eg; unknownMB”>
            <hasGraphics rdf: resource = “&eg; gamingGraphics”/>
        </rdf: Description>
    </hasMotherBoard>
</Computer>
<GamingComputer rdf: about = “&eg; alienBox51”/>
<Computer rdf: about = “&eg; binName42”>
    <hasMotherBoard rdf: resource = “&eg; bigNameSpecialMB”/>
    <hasBundle rdf: resource = “&eg; bigNameSpecialBundle”/>
</Computer>
```

基于两个关联文件创建推理器 (P169)

```
Model schema = ModelLoader. loadModel
                        (“file: data/owlDemoSchema.owl”);
Model data = ModelLoader. loadModel
                        (“file: data/owlDemoData.rdf”);
Reasoner reasoner = ReasonerRegistry. getOWLReasoner( );
reasoner = reasoner. bindSchema(schema);
InfModel infmodel = ModelFactory. createInfModel
                                (reaoner, data);
```

查找实例： nForce主板 (P170)

```
Resource nForce = infmodel.getResource("urn:x-hp:eg/nForce");
System. out. prioutln("nForce * :");
printStatements(infmodel, nForce, null, null);
```

```
public void printStatements(Model m, Resource s, Property p, Resource o)
{
    for(StmtIterator i = m.listStatements(s, p, o); i.hasNext( );)
    {
        Statement stmt = i.nextStatement( );
        System.out.println (" – "+PrintUtil.print(stmt));
    }
}
```

查找nForce主板的输出结果 (P170)

nForce * :

- (eg: nForce owl: sameIndivedualAs eg: unknownMB)
- (eg: nForce owl: sameIndivedualAs eg: nForce)
- (eg: nForce rdf: type owl: Thing)
- (eg: nForce owl: sameAs eg: unknownMB)
- (eg: nForce owl: sameAs eg: nForce)
- (eg: nForce rdf: type eg: MotherBoard)
- (eg: nForce rdf: type eg: Resource)
- (eg: nForce rdf: type a3b24: f7822755ad: -7ffd)
- (eg: nForce eg: hasGraphics eg: gamingGraphics)
- (eg: nForce eg: hasComponent eg: gamingGraphics)

```
<Computer rdf: about = “&eg; whiteBoxZX”>
    <hasMotherBoard rdf: resource = “&eg; nForce”/>
    <hasbundle>
        <GameBundle rdf: about = “&eg; actionPack”/>
    </hasbundle>
</Computer>
<Computer rdf: about = “&eg; whiteBoxZX”>
    <hasMotherBoard>
        <rdf: Description rdf: about = “&eg; unknownMB”>
            <hasGraphics rdf: resource = “&eg; gamingGraphics”/>
        </rdf: Description>
    </hasMotherBoard>
</Computer>
<GamingComputer rdf: about = “&eg; alienBox51”/>
<Computer rdf: about = “&eg; binName42”>
    <hasMotherBoard rdf: resource = “&eg; bigNameSpecialMB”/>
    <hasBundle rdf: resource = “&eg; bigNameSpecialBundle”/>
</Computer>
```

检验实例是否符合某个类表达式 (P171)

```
Resource whiteBox = infmodel.getResource("urn:x-hp:eg/whiteBoxZX");
if(infmodel.contains(whiteBox, RDF.type, gamingComputer))
{
    System.out.println("whiteBox recognized as gaming computer");
}
else
{
    System.out println("Failed to recognize whiteBox correctly");
}
```

```
ValidityReport validity = infmodel.validate( );
if (validity.isValid( ))
{
     System.out.println(“OK”);
}
else
{
   System.out.printIn(“Conflicts”);
   for (Iterator i = validity.getReports( ); i.hasNext( );)
   {
       ValidityReport.Report report = (ValidityReport.Report)i.next( );
       System.out.println(“ – ”+report);
   }
}
```

检验数据的一致性


课间休息


再会