# 结构工程语义网格本体模型框架的设计与实现

王 立 翁梓钧 苏 琳 邸瑞华
(北京工业大学计算机学院 北京 100124)

**摘 要** 针对结构工程领域中信息集成与共享时碰到的语义冲突问题,基于网格计算和语义Web的相关技术规范,设计和实现结构工程语义网格体系结构。并在此基础上,着重分析在构建结构工程语义网格语义层的过程中,在实例集成、模式集成、概念集成和查询应用几个阶段的语义冲突问题,设计与初步实现语义层本体模型框架。

**关键词** 网格计算,语义网,本体,信息集成
**中图法分类号** TP393 **文献标识码** A

## Design and Implementation of Ontology Model Framework in Structural Engineering Semantic Grid

WANG Li WENG Zi-jun SU Lin DI Rui-hua
(College of Computer Science, Beijing University of Technology, Beijing 100124, China)

**Abstract** Concerned with the semantic conflicts during information integration and share in structural engineering, based on the related technique criterions of grid computing and semantic Web, this paper proposed the architecture of structural engineering semantic grid, on this basis, the paper focused on analyzing the semantics conflict from instance integration, schema integration, concept integration to query processing during the course of building semantics layer of Structure engineering semantics grid, furthermore, the ontology model framework of semantic layer was proposed.

**Keywords** Grid computing, Semantic Web, Ontology, Information integration

随着模拟技术和传感器技术不断发展,在结构工程研究领域所产生的数据量正在呈现爆炸性的增长,对存储资源、计算资源、网络资源等都提出了极高的性能需求,为以往的数据资源管理技术带来了巨大的挑战。如何管理和使用这些数据资源,进而推动科学领域的研究,是当前迫切需要解决的问题。网格技术(Grid Technology)是20世纪90年代提出的新技术,它的目标是将各种分散的且隶属于不同管理组织的计算和数据资源,有机地结合起来,形成一个虚拟组织,从而实现组织内各种异构资源的透明共享和高效的人机协同工作。

但是,现有网格上存在信息格式异构、信息语义的多重性以及信息关系匮乏和非统一等问题,无法满足原先设想的网格应具有高度的简单实用和无缝自动化的需求,这是网格发展现状和应用需求之间目前存在的主要差距,因此需要在传统的网格中引入语义来改变这种局面。如何在更高的层次上更好地满足科学研究人员对数据管理与应用的需求,W3C(World Wide Web Consortium)针对智能设备对Web上存放的数据的理解和处理,提出了语义Web(Semantic Web)标准[1],充分利用知识工程和人工智能等相关领域取得的研究成果,建立了一系列的规范和技术标准(RDF标准和OWL标准等),对Web中存放的内容进行处理,目的在于使Web内容除了能被人类理解之外,还能被计算机和智能设备进行自动处理。

语义网格是语义Web和网格相结合产生的新的研究领域,它的出现为整个网格系统的语义互联提供了强有力的支撑,不仅能提供数据、计算服务和信息服务,而且引入知识层处理,提供知识的获取、使用、检索、发布等知识服务,体现了下一代网格的特点[2]。

本文针对结构工程领域中信息集成与共享的现状,基于CGSP网格中间件,并结合OWL和RDF等语义Web相关技术规范[3],设计和实现结构工程语义网格。并在此基础上,重点研究核心网格本体层的建模方案与相关技术。

## 1 建立结构工程语义网格面临的挑战

在信息集成领域,由于CORBA,XML和JDBC等数据交换协议的出现,平台和操作系统等异构问题已相对容易解决。然而随着研究的不断深入,人们对信息集成系统的需求不再局限于能够透明访问数据源,而是希望这些异源的数据能够在语义层上实现重组,进而满足用户的特定的语义需求。

目前,在构建结构工程语义网格过程中面临的挑战主要包括:

1) 结构工程领域术语标准的缺失

结构工程学科领域在不断演化发展,造成了领域术语的多样化,现有的术语标准不能满足结构工程信息化的实际需求,导致了现存的结构工程数据管理系统很少是按照这些标准来构建编制,很难对已建成的结构工程信息系统进行集成和共享,从而限制了数据挖掘和知识发现等各种高层应用的发展。

2) 语义数据集成的困难

实现异质异构数据的语义集成的前提是:用标准化的领

到稿日期:2008-07-11

域术语来描述的全局本体库和各个数据库具有一致的数据质量评估标准和维护策略[4]。在当前结构工程领域研究中，这些都不具备，导致了在语义集成中数据语义的丢失和脏数据规模的不可控。

3）异构系统之间的语义互操作

尽管 Web 服务为异构系统之间的互操作提供了实现基础，但是没有考虑异构系统在领域本体定义上的差异，从而无法解决异构系统之间语义互操作问题。

## 2 结构工程语义网格体系结构

本文提出的结构工程语义网格体系结构要求适应各种异质数据源的集成，并且能够快速灵活地应付数据源的变化。利用本体在描述语义上的优势解决数据集成中的语义异构问题。系统分为应用层、语义层、服务层和资源层，采用全局本体描述数据集成系统的全局模式，局部本体描述数据源模式，支持局部本体到全局本体的规则映射[5]。体系结构如图 1 所示。

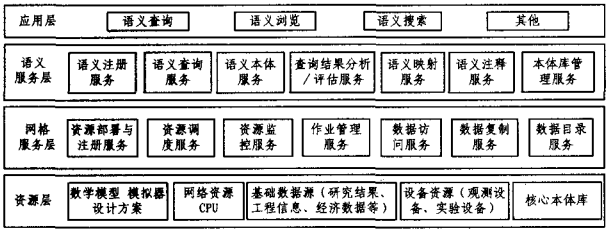


图 1 结构工程语义网格体系结构

1）应用层

该层位于系统的最顶层，直接与用户交互。主要包括两个部分：数据访问接口和显示界面。数据访问接口提供给用户基于语义的不同的数据访问方式，包括语义浏览、语义查询、语义搜索等。在此层操作的语义本体是基于全局模式描述的，用户不必关心底层数据源的分布情况和数据模式之间的差异。

2）语义服务层

该层是系统的核心部分，是在网格服务的基础上应用本体技术对不同结构、不同领域的知识进行统一语义定义，并通过服务实现对分布式的知识与数据进行统一的组织、执行以及协调，以便有效地管理语义、数据元以及可视化工具的语义描述。主要根据全局本体和局部本体的映射规则对用户的查询语句进行分解，最终生成面向各个局部数据源的子查询[6]。然后分发各个子查询任务到服务层，通过服务层提供的网格服务和 Web 服务实现对网格资源的访问。其作用是分解和执行应用层的基于语义的数据访问应用，并根据应用的具体需求，提供给下层的网格中间件，实现对相应的资源的调度运行。

3）基本网格服务层

该层是基于 CGSP 所提供的网格服务运行环境，利用和改造了 CGSP 中提供的网格公用服务。包括网格基础服务和面向结构工程应用的高级特色服务，介于网格资源与面向仿真应用的核心服务之间，根据上层对网格资源的需求，实现网格资源的部署/注册、调度、监控和目录服务等管理功能，同时包括对作业和资源数据等的核心管理服务。

4）资源层

提供结构工程语义网格调度使用的各类资源，包括理论研究中所用到的模型资源和计算资源，科研实验中用到的设备资源（观测设备和试验设备等）和基础数据资源（包括研究结果、实验数据和工程信息等），还包括结构工程本体库（定义了基本的概念术语以及概念之间的语义关系等）。

## 3 结构工程语义网格核心本体建模

在第 2 小节分析了在构建结构工程语义网格所面临的诸多挑战，本节在概念集成、模式集成、实例集成和查询应用等 4 个阶段考虑系统的语义异构问题，在此基础上，考虑语义层本体建模方案。

### 3.1 四阶段语义异构分析

在结构工程研究中，越来越多的数据资源被产生出来，且资源之间的依赖关系越来越复杂，导致需要整合的这些资源越来越多复杂。并且由于历史原因，高度自治的数据源彼此之间必然不可避免地存在语义层次的异构。语义的异构直接表现为各种不同类型的语义冲突。

1）标识符冲突

同一实体不同的标识引发的冲突。构件可以用 StuctureComponent 来表示，也可以用 Goujian 来表示。

2）结构冲突

相同实体在不同数据源采用不同的逻辑结构造成的冲突，包括：

模式同构冲突：属性集合不相似。系统 C1：Material (price, producer)；系统 C2：Materail(weight, shape)。

概括冲突：一个系统内多个实体被汇总到另外一个系统内的单个实体（此实体是局部源多个实体的概括），由此引发的结构冲突。C1(Steel, Wood)；C2(Meterial)。

拆分冲突：与概括冲突相反，外键的正确维护是关键。

聚合冲突：一个系统内单个实体的属性/值来自另外一个系统一些实体的属性/值的聚合。例如 C1 的派生数据的值由另外一些系统的原始数据经过傅里叶变换得到。

关系冲突：不同系统的相同实体与其他实体关系不一致引发的冲突。实验管理系统 C1：manage (Administrator, CommonUser)；数据管理系统 C2：Administrator 与 CommonUser 是对等关系。

属性描述冲突（属性的数据类型、长度）。

关键字冲突：针对关系模型中，含义相同的两个关系具有不同的关键字约束而导致的冲突。

约束冲突：不同系统对不同实体或关系采用不同的约束条件造成的冲突。C1：HealthOfBridge(Time, Material)；C2：HealthOfBridge(Time, Intensity)。

3）实例冲突

数据值表达冲突：同一数据值在不同领域、不同环境表达不同，在大量采用码表的系统中比较突出。数据值 Branch 在一个系统中代表“学科”，在另一个系统中代表“部门”。

数据表示冲突：YYYYDDMM 与 YYYY-DD-MM。

取值范围冲突。

数据量纲冲突：公制与英制。

数据精度冲突：例如，对实验结果进行评估：“百分制”，“优良差”。

数据可信度冲突：历史数据与实时数据的可信度不同。

目前，在语义 Web 和语义网格研究中，大都分立于不同阶段单独考虑系统中的语义冲突问题，但是，这不利于将语义问题统一规划处理[7]。本文从概念集成、模式集成、实例集成和查询应用这 4 个阶段分析语义异构问题，如图 2 所示。

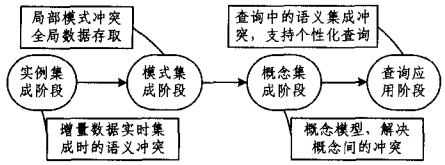


图 2　四阶段语义异构分析

在实例集成阶段，主要解决实例级的各类语义冲突问题，比如解决增量数据实时集中时的语义冲突。

在模式集成阶段，主要解决模式集成过程中语义冲突问题，如全局模式要(1)支持全局应用，(2)解决局部模式冲突和(3)支持全局物化数据的存储。

在概念集成阶段，要解决概念间的语义冲突问题，选择表达能力强的概念模型，并且能够反映局部概念的状态差异。

在查询应用阶段，主要解决查询中的语义冲突问题，包括查询时解决对未物化到全局的数据中存在的语义冲突和查询请求中的语义冲突，并支持个性化查询语义需求。

### 3.2　本体建模

针对 4 阶段中语义异构的分析，本文提出在语义层根据功能和特征对本体进行进一步的建模，如图 3 所示。

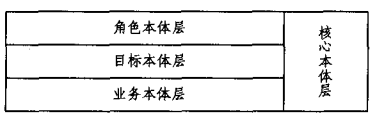


图 3　本体模型框架

1) 核心本体 CO(CoreOntology)

描述了结构工程领域中所使用的概念及其之间的关系。主要包括 3 个部分：属性描述(ModelProfile)，行为描述(ModelBehavior)和接口描述(ModelInterface)。按照逻辑功能分为两类：实体本体(EntityOntology)和操作本体(OperationOntology)。实体本体主要用来描述实体的概念和实体间的关系。操作本体描述如何操作或者使用这个实体概念和关系。本文采用 OWL(Web Ontology Language)语言来定义本体，采用 SWRL(Semantic Web Rule Language)语言建立规则描述本体之间的关系。

2) 角色本体 RO(RoleOntology)

通过定义角色本体可以分析组织架构，标示网格用户权限范围，并进一步决定业务目标。角色本体由一系列的职责 T(Task)组成，这些职责 T 与一些具体的业务活动 A(Activities)和规则 R(Rules)相关联。通过角色本体 RO，根据其职责 T 的不同，可以为不同角色的用户定制不同的偏好本体类，触发不同的推理规则和调用不同的业务逻辑流程，最终实现设定目标。

3) 目标本体 GO(GoalOntology)

目标本体 GO 基于 RO，它抽象于实际的业务目标。按照重要性划分，可以分为核心目标和一般目标。按照功能性划分，可以分为功能性目标和非功能性目标，非功能性目标一般不能被直接获得，它们可以是一些客观标准(如 QoS 指标)，功能性目标一般根据组织的需求和用户的偏好设定。按照复杂度划分，分为简单目标和复杂目标，复杂目标可以分解为一到多个子简单目标。此外，一个目标的执行可能依赖于另外一个目标的实现，同样，一个目标的实现可能导致另外一个目标无法实现。这种“依赖”与“对立”的关系可以通过设定不同的逻辑规则来实现。

4) 业务本体 BO(BusinessOntology)

根据实际的业务流程定义 BO，主要用来匹配服务层提供的网格服务或者 Web 服务，实现 GO 中定义的业务目标策略。BO 分为原子 BO 和合成 BO，每个原子 BO 由 4 部分组成，输入 Input、输出 Output、前置条件 Precondition 和后置条件 Postcondition、服务 QoS 信息等。Precondition 和 Postcondition 指定了开始执行 BO 和执行完成时的限制条件。BO 之间也可能存在依赖与排斥关系。通过 BO，可以在本体层建立抽象本体工作流，通过服务匹配机制，具体化为具体的服务流程执行。

### 3.3　应用案例

到目前为止，已经出现了很多本体建模工具，其中有些是独立于特定的语言可以导入和导出多种基于 Web 的本体描述语言格式。它们都是集成的本体开发环境或一组工具，支持本体开发生命周期中的大多数活动，并且因为都是基于组件的结构，很容易通过添加新的模块来提供更多的功能，具有良好的可扩展性。

本文采用 Protégé3. 3. 1 开发结构工程语义网格的本体模型(如图 4 所示)，基于 OWL-S(Web Ontology Language for Services)开发具体的语义 Web 服务，在语义层面上对服务接口、服务结构和服务交互等进行描述。

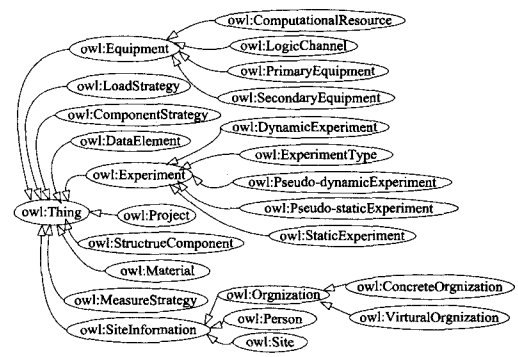


图 4　结构工程语义网格本体模型

利用建立的推理机模型 Infmodel 进行推理：

Property predicate= infmodel. getProperty(

“http://www. w3. org/2002/07/owl # equivalentClass”)；

Resourcesubject=infmodel. getResource(

“http://www. owl-ontologies. com/Material. owl # Material”)；

StmtIterator i = infmodel. listStatements(subjec, predicate, null)

每个关系由 3 元组(主语、谓语、宾语)组成，其中主语和宾语使用 Resource 表示，谓语使用 Property 表示，通过指定谓词和主语，可以推理出对应的宾语列表 StmtIterator。图 4

是对全局本体里的 Material 类进行相同概念的推理，其中谓语是 equivalentClass，主语是 Material，推理得到的结果放置在容器中，使用迭代器 i 就可以将结果顺序取出。

**结束语** 语义网格在网格迅速发展和普及的未来将起着越来越重要的作用。由于它可以理解用户的问题并且提供解决方案，它将会为终端用户提供更好更先进的服务。本文的研究对推动结构工程研究领域的信息集成与共享有着积极的意义。

当前的工作还处在初级的阶段，还有许多关键的问题需要进一步解决，如本体查询性能问题，目前采用的本体查询语言 SPARQL，其性能耗费远高于一般查询语言（如 SQL 和 XQuery），且当连接 SPARQL 和 XQUERY，或者连接 SPARQL 与 SQL 时非常复杂且花费巨大，这些都需要进一步研究、实践。

## 参 考 文 献


[1] Berners-Lee T, Hendler J, Lassila O. The Semantic Web. Scientific American, May 2001

[2] Trinh Q, Barker K, lhajj R. Semantic Interoperability between Relational Database Systems // 11th International Database Engineernig and Applications Symposium (IDEAS 2007). 200 IEEE

[3] Semantic Grid Community Portal. http://www.semanticgrid.org, 2004, 12

[4] 洪晓伟. 基于 XML 异构数据集成的研究[D]. 东南大学图书馆, 2004

[5] Goble C, Roure D D. The Semantic Grid: Myth Busting and Bridge Building. http://www.semanticgrid.org/docs/ECAISemanticGrid/ECAISe-manticGridFinal.pdf, 2004, 12

[6] 邓志鸿，唐世渭，张铭，等. Ontology 研究综述[J]. 北京大学学报：自然科学版，2002，38(5)：730-738

[7] Isabel F, Cruz Huiyong Xiao, Feihong Hsu. An ontology-based framework for XML semantic integration [C] // Washington: IDEAS'04 Workshop, Proceedings of the International Database Engineering and Applications Symposium IEEE computer Society. 2004: 217-226


(上接第 257 页)

列：$A=(a_1,a_2,\cdots,a_n)$，$a_i(1\leqslant i\leqslant n)$表示第 $i$ 个结点的左子树上的结点数目，$a_1,a_2,\cdots,a_n$ 是按二叉树 $T$ 的先序来排列的。

用上述相同的方法构造图 1(a)所示的二叉树，其数据结构表示如表 2 所列。

表 2 图 1(a)中的二叉树的数据结构

| | A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|---|
| Parent | Null | A | B | B | A | E | F |
| Lchild | B | C | Null | Null | F | Null | Null |
| Rchild | E | D | Null | Null | Null | G | Null |

算法如下：

```
For all pi par-do
  r=0;j<—i
If Lchild [j]! = Null then
  {r=r+1;j=Lchild [j]}
  Elseif j! =i then
  {If Rchild [j]! = Null then
    r=r+1;
  Endif
    j=Parent [j]
  }
 Endif
Endfor
```

## 4 实例分析

以图 1(a)所示的树为例，给出本文算法的实现过程。

第一步构造单链表，结果如图 2 所示。

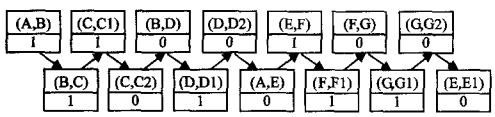


图 2 单链表示意图

第二步给单链表中的每个元素赋权值，结果如图 2 所示。

第三步计算单链表的每个元素的位序，每步计算所得结果如表 3 所列。

表 3 位序计算结果列表

| AB | BC | CC1 | CC2 | BD | DD1 | DD2 | AE | EF | FF1 | FG | GG1 | GG2 | EE1 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | |
| 2 | 2 | 1 | 0 | 1 | 1 | 0 | 1 | 2 | 1 | 1 | 1 | 0 | 0 | 1 跳 |
| 3 | 2 | 2 | 1 | 1 | 2 | 2 | 2 | 3 | 2 | 1 | 1 | 0 | 0 | 2 跳 |
| 4 | 4 | 4 | 3 | 4 | 4 | 3 | 3 | 3 | 2 | 1 | 1 | 0 | 0 | 4 跳 |
| 7 | 6 | 5 | 4 | 4 | 4 | 3 | 3 | 3 | 2 | 1 | 1 | 0 | 0 | 8 跳 |

第四步求结点的先序遍历顺序号权值为 1 的边有$(A,B),(B,C),(C,C1),(D,D1),(E,F),(F,F1),(G,G1)$，从而可以得到结点 $A,B,C,D,E,F,G$ 的位序分别为 7,6,5,4,3,2,1，由于结点个数 $N+1=8$，可以根据各结点的位序计算出各结点的先序遍历顺序号依次为 1,2,3,4,5,6,7，即这棵二叉树的遍历顺序为 $A,B,C,D,E,F,G$。

最后，根据第 3 节中 $A$ 序列的并行算法可求得 $A=(3,1,0,0,2,0,0)$。

**结束语** 一棵有 $N$ 个结点的二叉树，经改造后有 $2N+1$ 个结点的二叉树，有 $2N$ 条向下的有向边，因此，该算法需要 $2N$ 个处理机。在该算法中，求单链表中元素位序的计算时间复杂度为 $O(\log 2N)$，算法其余部分的计算时间是常数，所以该算法的时间复杂度为 $O(\log 2N)$。

## 参 考 文 献


[1] Ahrabian H, Nowzari - Dalini A. Parallel generation of binary trees in A-order. Parallel Computing, 2005, 31: 948-955

[2] Ahrabian H, Nowzari - Dalini A. On the generation of binary trees in A-order. Int. J. Comput. Math., 1999, 71: 1-7

[3] 严蔚敏，吴伟民. 数据结构(C 语言版). 北京：清华大学出版社，1999

[5] Ahrabian H, Nowzari - Dalini A. Parallel generation of binary trees in A-order. Parallel Computing, 2005, 31: 948-955

[4] Korsh J F. A-order generation of K-ary trees with 4K-4 letter alphabet. J. Infom. Optim. Sci., 1995, 16(3): 557-567