申请上海交通大学工程硕士学位论文

基于本体面向三维场景的语义标注研究与应用

学校代码：10248
作者姓名：施梦炜
学　　号：1100379035
第一导师：姜丽红
第二导师：
学科专业：软件工程
答辩日期：2013 年 1 月 9 日

上海交通大学软件学院

2012 年 12 月

A Dissertation Submitted to Shanghai Jiao Tong University

for Master Degree of Engineering

STUDY AND APPLICATION OF ONTOLOGY-BASED APPROACH TO SEMANTIC ANNOTATION OF 3D SCENES

University Code：	10248
Author：	Shi Mengwei
Student ID:	1100379035
Mentor 1：	Jiang Lihong
Mentor 2:	
Field：	Software Engineering
Date of Oral Defense：	2013-01-09

School of Software

Shanghai Jiaotong University

December, 2012

上海交通大学

学位论文原创性声明

本人郑重声明：所呈交的学位论文，是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的作品成果。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律结果由本人承担。

学位论文作者签名：

日期：　　年　　月　　日

上海交通大学

学位论文版权使用授权书

本学位论文作者完全了解学校有关保留、使用学位论文的规定，同意学校保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权上海交通大学可以将本学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本学位论文。

保密□，在___年解密后适用本授权书。

本学位论文属于

不保密□。

（请在以上方框内打“√”）

学位论文作者签名：　　指导教师签名：

日期：　　年　　月　　日　　日期：　　年　　月　　日

基于本体面向三维场景的语义标注研究与应用

摘 要

随着计算机辅助设计，三维虚拟化技术的快速发展，在它们的实际使用中产生了大量的三维场景和模型。针对它们的精确检索和分类因此成为有效利用这些数据的关键所在。语义标注技术在一定程度上解决了三维知识管理的难题，但在应用数据越来越趋向海量的今天，其本身也面临着新的挑战。这些挑战包括：如何缩短标注时间，减少标注成本，使其面对数据增长具有可扩展性；如何降低对领域专家的依赖，使其趋向自动化；如何维护标注的一致性，抑制内容相互冲突的标注；以及如何在做到上述几点的同时，仍然保证标注内容与手动完成的相比同等的准确性。

本文以传统的语义标注技术为基础，进行针对标注半自动化过程的优化，来解决现有语义标注过程中重复劳动多，效率低下的问题。本文关注于网络三维文档，并根据其特点提出多侧面的本体模型，对三维场景建模中各领域通用的概念进行抽象以及复用；同时提出标注规则，帮助标注者抽象标注方法，以达到标注过程半自动化以及标注内容一致性可验证的目的。本文最后通过案例验证由文中提出方法所完成的标注和传统的手工标注比较起来，拥有同等的准确性，可以在后续的查询检索操作中起到有效的作用。

完成的工作有：

一、面向网络三维场景，提出一套基于本体和规则的半自动化语义标注方法。给出完整的方法流程，实现框架以及应用场景。

二、提出针对三维场景文档的本体建模。给出用于描述三维场景建模基本概念及属性的通用本体的实现方式。在此基础上，描述从网络三维文档到本体实例的映射算法，该算法能够从三维文档中抽取物体的几何，空间关系等可视属性。

三、提出标注规则的语法和语义，能够灵活地表达特定的标注模式。在此基础上，完成标注引擎的实现算法，包括规则一致性检测，规则在文档中的匹配，以及规则的应用方法。

四、设计了面向网络三维场景基于本体和规则的半自动化语义标注原型系统，并在

实际应用场景中验证标注过程半自动化的有效性和准确性。

本文所提出的半自动化语义标注方法，基于传统使用本体的语义标注，继承了后者描述精确的特点。此外，通过对它进行改进，实现了利用规则进行标注逻辑抽象，从而进一步达到半自动化的目标。本文还验证了半自动生成的标注内容并没有比纯手动标注降低准确性。因此，该方法有效地提高了网络三维文档的标注效率，在大量数据的情况下，能避免标注过程中过多的重复工作，快速地应对新增的文件，是后续使用中全面的查询与检索操作的基础支撑，对三维场景的语义管理具有较高的参考价值。

关键词 语义标注，三维场景，X3D，本体

STUDY AND APPLICATION OF ONTOLOGY-BASED APPROACH TO SEMANTIC ANNOTATION OF 3D SCENES

ABSTRACT

With the rapid development of computer aided design and 3D virtualization, large numbers of 3D models and scenes have been created during the application of those technologies. The key to the effective using of these data lies in the ability to perform accurate search and classification on them. The technique of semantic annotation has to some extent solved the problem of knowledge management on 3-dimensional data while facing new challenges as the scale of documents keeps increasing. These challenges include how to decrease the time cost of annotation process in order to make it more scalable to the increasing of data sizes; how to reduce the degree to which the annotation process has a dependency on domain experts in order to automate the former; how to maintain the consistency of annotation contents, thus limiting those that conflict with each other; and how to guarantee the correctness of annotation content compared to those manually created given that the previous aims have been achieved.

In this paper, a traditional semantic annotation based approach is proposed to provide an optimization for the semi-automatic semantic annotation, in order to solve the problem of redundancy of repeated human work and low efficiency in the process of semantic annotation. This paper focuses on X3D document and proposes multi-aspect ontology model according to its specification to abstract and reuse the basic concepts in 3-dimensional modeling over

various domains. Besides, this paper proposes annotation rules for annotators to abstract annotation processes to achieve the goal of semi-automatic annotation and verification of the consistency of annotation contents. This paper concludes the validity of the approach by using it in a scenario and comparing the generated annotation contents with the manual-created ones to demonstrate that they have the same level of correctness and thus the former can be effectively used in coming query and search operations.

The main research work is as follows:

(1) A complete approach of ontology-based, 3-dimensional scenes oriented, semi-automatic semantic annotation, along with its working process, implementation framework and application scenario is proposed

(2) Ontology modeling directed at 3-dimensional documents is provided. A general ontology that describes the basic concepts and attributes of 3-dimensional scene modeling is implemented. Based on that, algorithms that map visual information in 3-dimensional documents to ontology axioms are proposed. The algorithms are responsible for the extraction of visual information from the 3D documents.

(3) The representation and semantics of annotation rules are defined. Efficient algorithms to search for patterns of rules that can fire among large numbers of documents are implemented. Mechanisms to verify the consistency of rules are proposed.

(4) A prototype system is implemented to validate the effectiveness and correctness of semi-automatic annotation within a particular scenario.

The annotation method proposed in this paper is based on the traditional method which is ontology based, and inherits the feature of accuracy of the latter. Moreover, this method achieves the goal of semi-automation via improvement to the original one and abstraction of annotation logic using rules. It is verified that by automating the correctness has not been decreased compared to pure manual annotation. Therefore, the method reduces repeated work, is scalable to new documents and proves to be valuable in semantic management.

Keywords Semantic Annotation, 3D Scenes, X3D, Ontology

目 录

1 绪 论

1.1 研究背景与意义

近年来，伴随着三维数字化技术，虚拟现实与计算机辅助设计/制造的发展与普及，三维模型及场景得到越来越多的欢迎。三维文档的数量也因此而快速增长。然而如果没有成熟的软件进行支持，这些模型将在很多程度上只能作为纯粹的几何物体存在，从而限制了它们在设计中能起到的作用。基于以上原因，针对三维模型或场景的应用对底层的管理软件提出了更高的需求。首先，这些软件需要作为搜索引擎，能够在大量的三维数据中检索满足指定条件的模型或者场景；其次，它们必须能够迅速地抽取出这些模型的三维结构，来支持例如语义查询，或者构建更为庞大的三维场景这样的高层应用。

这些需求的提出直接引发了对三维模型进行语义标注的想法。所谓语义标注，是指模型的提供者除了存储三维模型或者场景的几何特征及结构信息之外，同时还要维护它们的语义，譬如某个物体所属的类型，以及它在实际工程中用途等，并且将前后两者以某种方式关联起来。为了满足上文所提出的诸多后续应用要求，标注过程必须满足以下几个条件：

首先，标注知识的载体必须具有一定的精确性。它所代表的标注内容能够精确而无二义性地表达一个领域知识。该条件是为了保证在后续的查询检索等应用中，用户可以准确地找到自己想要的三维场景，同时不会看到太多与查询条件无关的场景。

其次，标注过程需要具有一定的高效性。若标注过程本身是一项繁琐而耗时的工程，则在它之上消耗的时间和人力最终将无法支撑其之后在检索上提供的优势和便利。这里指的高效性包括两个方面：其一，需要有计算机软件或工具支持来保证标注工作可以以便捷的方式进行，同时标注内容能够得到快速的查询和存取；其二，标注过程中的重复行为能够得到识别，并以某种方式得到自动化的执行。

第三，对于任何程度的自动标注过程来讲，三维模型或场景中的基本视觉信息需要得到识别，并能够和它们在更高层上所表示的语义知识产生联系。其原因在于三维场景的基本组成结构无外乎简单物体和其色彩材质等信息的叠加，在这些内容中却包含了实际应用领域中需要的语义信息。因此在它们之间存在着一定的转化关系，可以通过挖掘而被利用。

基于以上需求，本文提出了一套基于本体面向网络三维文档的半自动语义标注框架，旨在保证标注正确性的前提下，利用计算机软件系统根据用户指定的规则针对三维场景文档进行自动标注。本文提出通用本体来抽象三维场景建模中的视觉信息，并提出算法来抽取该类信息。在此基础上，本文构建标注规则和规则实施引擎来实现标注过程的半自动化。

1.2 国内外研究现状

(1)半自动的语义标注方法

语义标注是为了实现语义网[1]以及其他基于语义信息的应用的基础。语义标记的目标是将本体实例数据与本体的类相互关联起来。手动的语义标注在现今的研究背景下已经变得十分容易，用诸如 OntoMat[2]之类的软件工具就可以轻松完成。然而，使用人工标注也常常伴随了许多问题的出现。因为人工操作可能是许多错误的来源，这其中包括标注者对业务领域的不熟悉，标注者接受训练的程度不足，以及待标注文档的复杂度过高以至于标注者无法胜任。另一方面，人工标注也因为成本太高而无法应用于海量数据的标注工程中。自动或者半自动的标注过程随即成为解决这一问题的唯一途径，尤其是半自动化的标注方法，得到了大量的研究及引用，因为全自动对计算机的智能要求较高，在现阶段往往无法取得较好的效果[3]。

关于（半）自动化进行语义标注的方法存在三种，分别是监管条件下基于机器学习的方法；无监管条件下基于机器学习的方法[4]以及基于规则的标注方法。监管环境下基于机器学习的方法包括了两个阶段：标注阶段和训练阶段。在标注阶段中，输入数据是结构化或者半结构化的文档，目标是识别这些文档中的实体以及它们之间的关系。在训练阶段中，实际的任务是学习将在后续标注过程中被使用到的标注模型。

D.Xu 等人使用了神经网络的方式来进行概念的学习[5]。他们使用了 Princeton Shape Benchmark(PSB) [6] 数据库中的若干分类来进行手动标注，然后将后者送入神经网络的分类器。通过这种方式，文档内的底层信息与高层语义知识之间的关系被建立了起来。这种学习方式的缺点是比较耗时，并且需要一开始拥有大量的训练集数据。

R. Ohbuchi 领导的研究提出了一种半监管环境下的学习方式，他们首先利用无监管的学习将一堆未标记的三维模型根据它们的特征分类到若干子空间中，随后在各自的小范围内利用监管学习来进行更细致的分类。他们的实验结果显示这种做法比单纯监管或无监管的方式要更加有效[7]。

另一方面，基于规则的自动标注方法也被广泛引用，因为该方法在提供一定自动程

度的同时，还因为标注规则是领域专家对标注过程的一种抽象，因此具有较高的准确性。典型的研究如 Klien 的工作，他们建立了标注规则来在大量的地理数据中寻找符合一定特征的数据子集，并将它们映射到各自的地理概念上，为地理空间语义网的实现提供了坚实的基础[8]。另外，Djioua 等人提出了 EXCOM 系统[9]。该系统同样致力于实现语义网，着眼于在大量网页中进行标注。他们提出的规则主要利用到网页的标注以及正则表达式，针对于文章的段落，标题以及文字等信息。

(2) 基于本体的语义标注方法

语义标注在最初应用的时候往往使用关键字作为标注知识的载体。基于关键字的标注方式虽然因为自然语言的不精确性和随意性具有一定的歧义，在检索过程中准确率较低，但是从普通用户（非专业领域使用者）的角度来讲，却是十分适合的做法。例如 Flickr[10]允许用户上传相片，并且随意地对它们贴上标签。类似的，stackoverflow[11]等系列网络知识共享站点允许用户在进行提问的同时，对所提出的问题进行类别的划分。从上述例子中可以看出，基于关键字的标注方式在 web2.0 环境下多用户的协同标注中具有广泛的应用。

然而在其它更具有专业性的领域，基于关键字标注的弊端使得它们的可用性大大降低。为了解决该问题，很多研究提出了利用本体作为知识载体的标注方式。本体是一种针对一系列共享概念的详细的形式化表示方式。本体利用利用其形式化的严格性，使得它能够被应用于计算机逻辑推理而在标注中获得更多有用的知识。

在利用本体进行语义标注的研究中，B. Popov 等人总结了利用本体和自然语言处理在媒体新闻中抽取人事和企业信息的一般做法[12]。在 L. Hollink 等的工作中，他们建立了一套表示画作和画家等的本体，并利用它对某展览馆的作品进行了标注。他们建立了软件接口来支持用户针对画作名称，画作风格等信息的检索[13]。该研究的缺点是，他们无法对画作中的一部分进行具体的标注，从而欠缺了图像的内容检索。作为该缺点的改进，J. Assfal 将特征抽取方法应用到体育比赛的视频中，来识别比赛中的关键事件，并对其进行自动标注[14]。S. Bloehdorn 等的研究则采用了图像描述符的手段来匹配图像中的特殊区域，并对它们进行标注[15]。

(3) 针对三维场景的标注方法

针对三维模型和场景的语义标注问题，最重要的是建立图形与它们所表示的语义之间的桥梁[16,17]。为了建立这个连接，许多研究都尝试对三维场景的内容进行分析，来挖

掘三维视觉信息之中所携带的语义。在基于内容的标注中较为著名的是 M. Attene 等人的研究[18]。他们的研究使用到了模型分割算法将一个三维模型识别出不同的组成部分，并对其中的每一个部分检查是否符合预定义的几何特征。对于符合那些几何特征的组成部分，将它们映射到对应的本体类别以及本体属性之上。另外，王新颖等人实现了一种基于本体的三维模型检索系统，能够根据检索条件，推理和查询知识库内模型的语义信息[19]。

除此以外，另有很多研究的目标如本文一样直接定位在 X3D[20,21]之上。X3D 是一种专为万维网而设计的三维图像标记语言，由 Web3D 联盟设计的，是 VRML 标准的最新的升级版本[22]。在利用 X3D 做实际应用的标注研究中，X. Cao 等人提出了在不同用户间利用 X3D 进行协同标注，并复用标注知识的框架[23]。F. Pittarello 等的研究提出了一种标注方式，他们不仅考虑将 X3D 的物体节点进行语义标注，同时还考虑将场景内的空间进行一定程度的标注[24-26]。这项研究的另一个分支致力于使用户以最灵活的方式在 X3D 之上进行标注[27]，并提出了一个系统将这两者给结合起来[28]。然而这种方式的缺点是，空间引入的语义信息并不是三维场景中唯一一种丢失的语义信息，单独考虑空间语义太过于具体化，如果要加入别的语义，则需要重新进行相关标注方式的研究，最佳情况下也只是利用之前的研究成果进行复用，而无法完全抽象出语义标注的过程。同时该标注过程只能是手动的，当遇到大量场景时，标注标率会有所下降。

1.3 主要研究内容

综合上述研究现状及现在的研究热点，本课题将提出一套针对三维模型及场景基于 X3D 文件格式的半自动语义标注框架，来实现在保证标注内容正确性的前提下对标注过程进行效率的提升。主要的研究内容包括以下几点：

其一，对三维场景建模的语义标注进行本体建模。这其中包括用于表示三维场景建模中基本视觉信息的通用本体；用于在标注规则中对文档进行描述的文档本体；以及最终标注内容将要使用的领域本体。提出通用本体和文档本体所要满足的条件，具体实现，以及它们和领域本体的相互关系。

其二，提出算法来抽取三维场景文档中的视觉信息。这其中包括：分析并识别出三维文档中视觉信息的种类；针对每一种视觉信息提出将它们从三维文档中抽取出来的策略；利用通用本体来表达这些抽取出来的知识。

其三，建立基于规则的半自动标注引擎。这其中包括：提出标注规则语言来表示标注过程；提出标注规则一致性检测的算法；构建合适的数据结构来表达用户定义的规则，使得标注引擎能够在大量三维场景文档中快速找出符合规则的部分，并将所牵涉到的规

则付诸实施。

最后，构建原型系统。实现上述思想和算法，搭建基于本体面向 X3D 场景的半自动语义标注软件原型系统。该系统中包含三维场景视觉信息提取，针对三维场景进行手动标注，通过解析规则和手动标注进行自动标注等功能。使用一个具体应用场景实施上述流程，验证本文提出的系统的实用性。

1.4 论文组织结构

第一章“绪论”部分主要介绍本文的研究背景和国内外研究现状。

第二章“面向 X3D 的语义标注框架”通过应用场景分析介绍本文所提出的方法的主要框架和其涉及到的模块。并简单介绍标注结果的持久化形式。

第三章“三维场景文档的本体描述”提出本文用到的通用本体，文档本体和领域本体的概念。介绍了通用本体需要满足的条件，并提出一个通用本体的实现。最后提出从三维文档中抽取可视化信息的算法。

第四章“基于规则的语义标注研究”提出了本文用到的标注语言，标注规则的匹配和应用算法。

第五章“系统实现与验证”讨论了本文研究中所实现的原型系统的实现细节。并在此基础上用一个例子演示了完整的标注流程。

第六章“总结与展望”总结了本文的主要工作，并对未来可能的研究方向做出了展望。

2 面向 X3D 的语义标注框架

本章将提出面向 X3D 的半自动化语义标注过程的总体框架。首先，我们将通过若干应用场景的分析来引导出标注框架的需求，并据此提出本文所研究的标注框架。随后，本章将介绍该框架内执行三维场景语义标注的一个典型流程。框架中的主要部件及它们所起的作用将在这之后得到简要的描述。最后，本章将简述标注内容的持久化表示结构及方法。

2.1 应用场景分析

近年随着三维文档的受欢迎程度不断提高，越来越多的应用中涉及到了针对三维场景的查询和检索。一个常见的应用场景例如一个使用工业零件三维模型库的制造业企业，它希望能根据特性或参数，精确地从大量的零件中找到匹配的模型，从而指导并完成其生产线上的加工，拼装等流程。该应用提出了这样的需求：首先，库中的每个模型都要携带额外的数据来描述它们的特性或参数；其次，这些信息必须能够精确地反应其所对应模型的领域信息。

另一个应用场景来源于一个家具装潢电子商务网站的用户。他们希望能够从该网站提供的大量居室场景中，根据其内部的某些细节特征，找到符合条件的房屋。这些查询请求可能包括房屋内地板的质量，某一件房间的采光条件，或者居室间的相互连接。该应用场景要求：场景文档所附加的语义信息不仅能够描述场景作为一个整体，还要能描述场景内部的组成部分及其相互关系。

为了满足这些应用需求，三维模型场景库的创建者或供应商需要对他们的三维文档进行语义标注工作。图 2-1 展示了语义标注的过程和它们的最终应用。在该图中，三维文档经过标注流程而产生了语义标注信息，这些信息被存储起来，并成为检索软件系统的知识库，为用户提供精确高效的查询服务。

对于标注工作来讲，它则需要满足以下几个要求：首先，标注所产生的信息需要以一种精确的方式来表达，从而保证用户的查询条件和他得到的结果都具有准确性；其次，三维模型场景中的基本视觉信息和物体间空间关系需要得到识别，并能够和它们在更高层上所表示的语义知识产生联系；最后，标注过程本身需要具有一定的高效性。对于大量的三维模型场景，标注人员将不需要为每一条标注内容进行手工输入。他们的部分工

作需要由标注系统的自动化来代替。

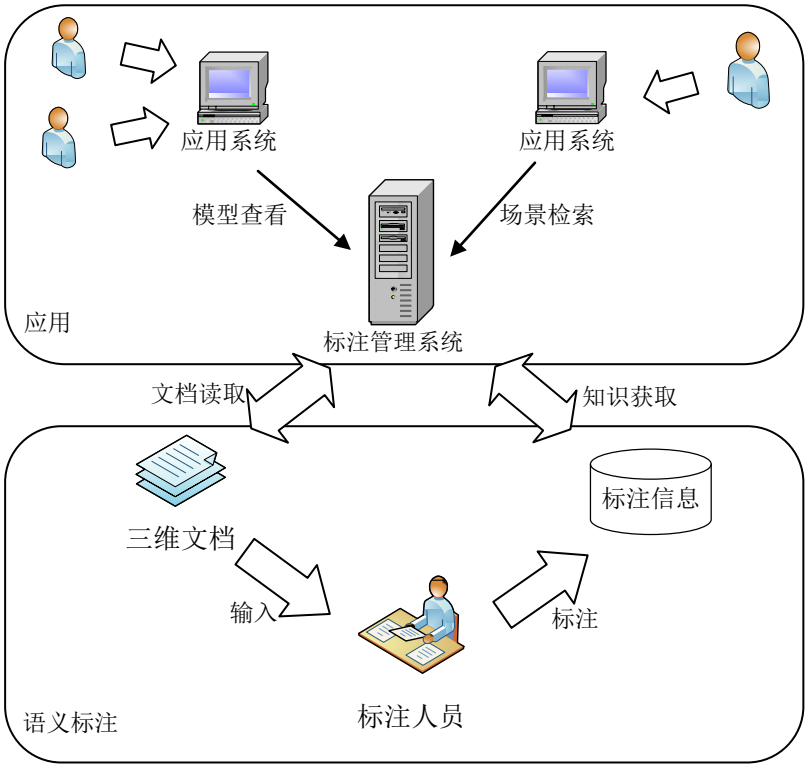

图 2-1 语义标注过程与其应用场景

Fig.2-1 Semantic Annotation and Its Application Scenario

2.2 面向 Web3D 的半自动化语义标注总体框架

本文所提出的半自动化语义标注系统旨在不影响标注内容正确性的前提下，将部分标注转化为可由计算机程序自动生成，从而减少手工重复工作。它的主要框架如图 2-2 所示。其中，由灰色阴影标识的部分标识由计算机自动处理完成，其余则需要人工干预。该图由若干部件以及它们之间的连线组成，连线上方的数字表示在一个典型的标注流程中，各步骤的执行次序。

2.2.1 标注的预先准备

在任何标注工作得以开始进行之前，我们假设图中的以下两件工件是已经完成并可付诸使用的。第一是通用本体和文档本体。通用本体代表在三维场景建模中通用的知识，比如物体形状以及空间关系等概念。文档本体用于描述文档中的节点关系，它主要在规则的定义中被用到。我们将在第三章详细介绍通用本体需要符合哪些要求，并示例性地

给出一个通用本体的实现方式。

第二是用 X3D 格式表示的场景或模型文件。这些文件是我们进行标注的对象，我们假设已经存在若干这样的文档等待标注。每个待标注的文档都拥有一个唯一标识的字符串，通常这个字符串是该文档在某文件系统中的路径，使得日后我们通过标注信息检索的时候，可以快速的定位到并提取出相关的文件。由 X3D 格式构建的三维场景是本研究的输入内容之一。本研究选择 X3D 作为标注对象的原因主要有以下几条：

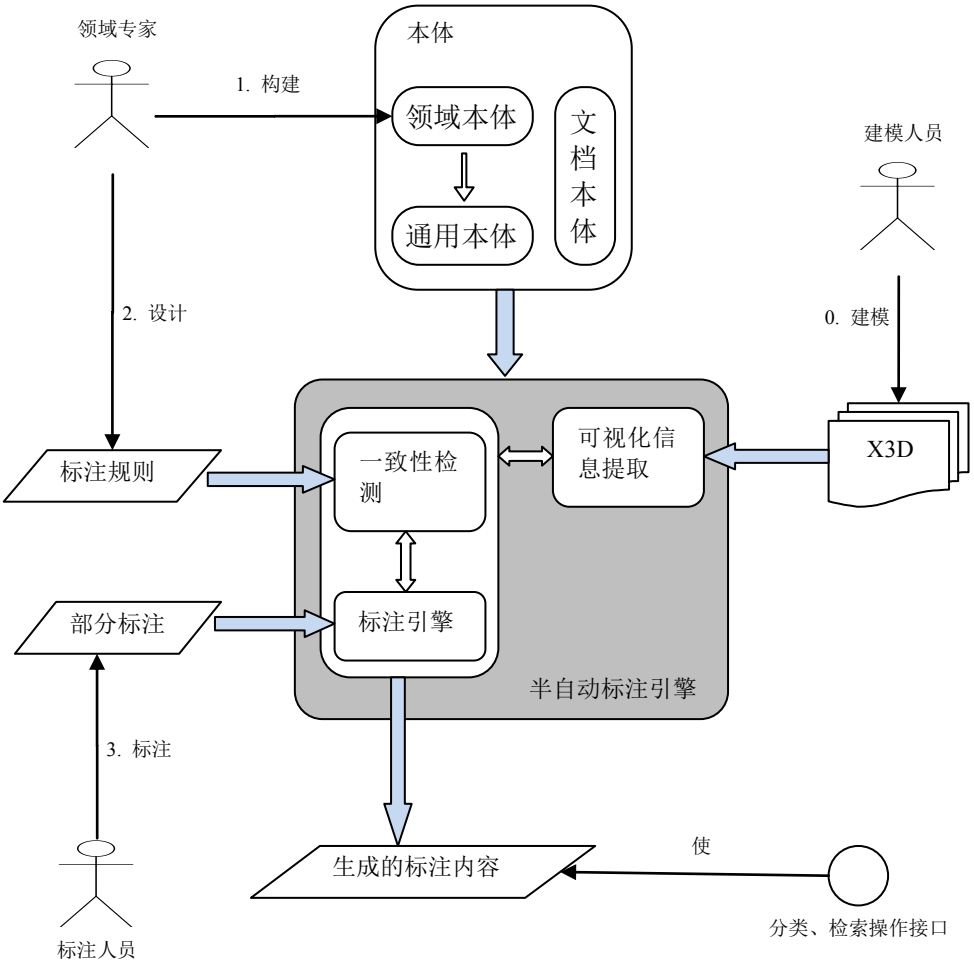

图 2-2 半自动语义标注总体框架

Fig.2-2 General Framework of Semi-Automatic Semantic Annotation

1. 该格式存在基于 XML 的文件格式，其层级结构有利于当今绝大多数编程语言的处理。此外其文件的层次结构和其所表示的场景内容的层次结构有着良好的对应关系，在实际处理中具有优势。

2. X3D 中以自顶向下的方式来描述场景。这使得在实际处理中能够方便地分离不同的物体，以及获取旋转平移等关系。在极端情况下，该文件格式并不避免建模人员以

纯多边形面片的形式构造一片场景。在这种特殊情况下，本文所提出的标注过程所能提到的作用将会非常有限。但是另一方面，这样的建模方式也会使得 X3D 本身的优势在一开始就无法体现。因此，良好的建模方式在本文中存在两层意义：第一，它是半自动标注方式得以起到较大作用的必要条件；第二，一旦此条件得以满足，我们即可认为建模人员在创建场景的时刻已经通过其自顶向下的层次结构为文档中注入了一定的语义信息，而本文所要做的正是将这一部分语义重新挖掘出来。

3. 在 HTML5 得到越来越多推广的同时，我们可以预见 X3D 将在不久的将来得到越来越多的应用。X3D 所独特定义的动画，事件等内容也将可能成为语义标注的对象。大多数在本文中所举的例子将不会涉及到这些特性，但是本文所提出的方法却可以快速的进行扩展，以适应和应对这些需求。在第三章中这一点将得到简要的阐述。

2.2.2 一个典型的标注流程

当上一小节提到的工件准备完毕后，本文提出的标注流程将通过以下步骤一一进行（该步骤的顺序对应于图 2-2 中箭头所关联的顺序）：

1. 由领域专家建立领域本体。该领域本体的内容不再是三维建模的通用知识，而是相关领域的实际概念。例如在家居装潢领域，这些概念可能包含桌椅等各类家具。

2. 由领域专家建立标注规则。这些规则将场景中的模式映射到标注内容，用于告诉标注引擎拥有特定特征的物体将获得哪些标注信息。标注规则在本文中被看做标注方法的一种抽象，它是标注人员对自己手动标注过程进行整理思考后，总结出的关于相似标注的操作模式。我们将利用这种模式的重复性来进行标注过程的优化。

3. 针对领域专家制定的标注规则，由标注引擎进行规则一致性的验证，也即检测规则集合中是否存在互相冲突的规则。

4. 对于待标注的三维场景文档，由标注人员进行部分标注。这些部分标注代表了需要启动自动标注的最少条件。通过这些少量的标注与标注规则的结合，标注引擎将能够推理出更多的标注内容，从而省去了许多的人工操作。

5. 由自动标注引擎结合上述步骤的产物，进行文档的模式匹配以及规则应用等工作，完成自动标注。这些标注内容（包括在步骤 4 中提到的人工完成的标注）将作为场景文档在后续使用中分类以及检索的重要信息。

对于相同的领域，上述过程中的前两步将不需要重复进行。也就是说，对于同一个领域，已经完成的领域本体以及标注知识将可以得到复用。对于新产生的场景文档，我们只需要对其做最少量的标注，之后的工作将交给自动标注引擎。当然，对于不用的领

域，用户仍旧需要为它们各自建立不同的领域本体及规则，或者当领域性质相近时，可以修改现存的某些本体及规则，从而达到部分复用的效果。

如果我们将本体视为系统中统一使用的描述语言，则整个标注过程可以看作是运行在一个以 X3D 场景文档和标注规则为输入，以标注内容为输出的系统内。该过程的流程图如 2-3 所示。

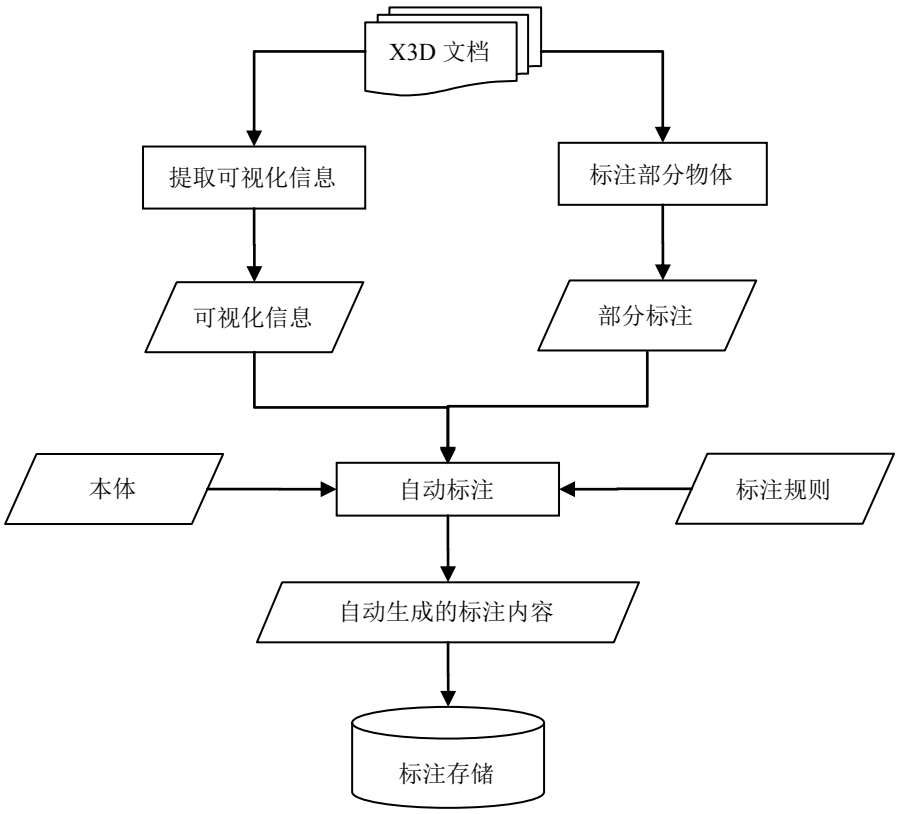

图 2-3 标注过程流程图

Fig.2-3 Flow Diagram of Annotation Process

从该流程图中可以看出，X3D 文档将首先（分别经由人工和标注引擎中的可视化信息提取模块）得到部分标注和可视化信息，随后由自动标注引擎在标注规则的指导下对这些信息进行处理得到生成的标注。这些生成的标注将被存储在数据库中，用作将来检索和查询的依据。

2.2.3 标注过程的主要模块

从图 2-2 中可以看出，自动标注引擎的内部结构较为复杂，以下针对其中的主要模块做简要的介绍。它们和整个框架的其他部件一起协作，保证了本文提出的标注方法可

以快速并正确地生成基于 X3D 文档的语义标注内容，从而达到标注过程效率提升的目标。

1. 可视化信息提取模块。该模块负责处理场景文档中物体的几何以及外观信息，并将他们统一转换成由通用本体所描述的形式。在处理前，这些信息在文档中的表述可能缺乏标准的形式，并散落在文档的各处。处理后这些信息将得到统一的描述，便于后续的比较和匹配。具体用来提取这些信息的算法将在第三章中阐述。

2. 一致性检测模块。该模块负责检验用户所定义的规则中是否存在相互冲突的地方。本文主要侧重于两方面的冲突：一是同一个标注规则下标注信息在本体意义上是否存在冲突，二是不同的标注规则间是否会存在循环触发标注的情况。该算法将在第四章中得到描述。

3. 标注引擎。该模块是标注过程中最核心的模块，它负责对文档中的物体特征进行搜索，找出匹配标注规则的部分，并应用规则来自动生成标注信息。关于它的具体架构和实施细节同样将在第四章中描述。

2.3 标注结果的持久化表示

标注内容是本文提出的标注引擎的最终输出。本文使用本体作为知识的表述形式。在本文研究中，有两种方式可以产生知识，分别是由图 2-2 中步骤 3 标注人员完成的部分标注，以及由标注引擎自动生成的标注。这两种标注的表示形式根据它们作用的不同将分为以下两个类别：

第一个类别的知识表示三维文档内物体和本体实例的关联关系。具体来讲，每一个属于该类别的知识都将以一个三元组的形式存在。表 2-1 给出了该组合中每一个成员的含义以及它们的示例。

表 2-1 文档与本体关联的标注结构及示例

Table2-1 Structure and Example of Annotation about Ontology-Document Relations

标注结构	示例
文档描述符	file://computer-name/x3d-documents/furniture/chairs/chair-1.x3d
物体描述符	/X3D/scene/transform[1]/transform
本体实例	www.company-name.com/furniture-ontology.owl#chair-1

在该组合中，首先文档描述符描述了某个 X3D 文档的具体位置，在整个标注系统中，该路径唯一标识了该文档。我们在示例中给出了一个典型的文件路径。其次，物体

描述符则描述了在某个 X3D 文档内，一个特定物体在文档结构中的位置。我们的示例中给出的是一个 XML 文档的节点路径的描述字符串，通过解析这个字符串可以在相应的 XML 节点树中找到其对应的节点。最后，本体实例唯一的标识了一个本体知识库中的某个实例。在我们的示例中，本体实例表现为一个本体的 URI，之后，针对该物体的属性等标注信息都将通过与其关联的本体实例来描述。

第二个类别的知识表示了具体每个物体所持有的属性，例如一间房间的防火级别。这些属性同样以三元组的形式存在，例如上面的例子可以表示为表 2-2 所示的知识条目。该条目记录了 room-1 实例拥有 low-fire-protection 的防火级别。

表 2-2 本体属性关联的标注结构及示例

Table2-2 Structure and Example of Annotation about Ontological Properties Relations

标注结构	示例
本体实例	www.company-name.com/furniture-ontology.owl#room-1
属性	www.company-name.com/furniture-ontology.owl#has-fire-protection
属性值	www.company-name.com/furniture-ontology.owl#low-fire-protection

标注内容与场景文件的对应关系如图 2-4 所示。图中由灰色区域包含的即为标注内容的实际存储区域，它包含两种结构，从上到下分别为上面所描述的两类别知识。第一类结构将文件中的相应内容对应到场景中的物体，而第二类结构则用本体的类和属性来标注某个实例的具体特征。

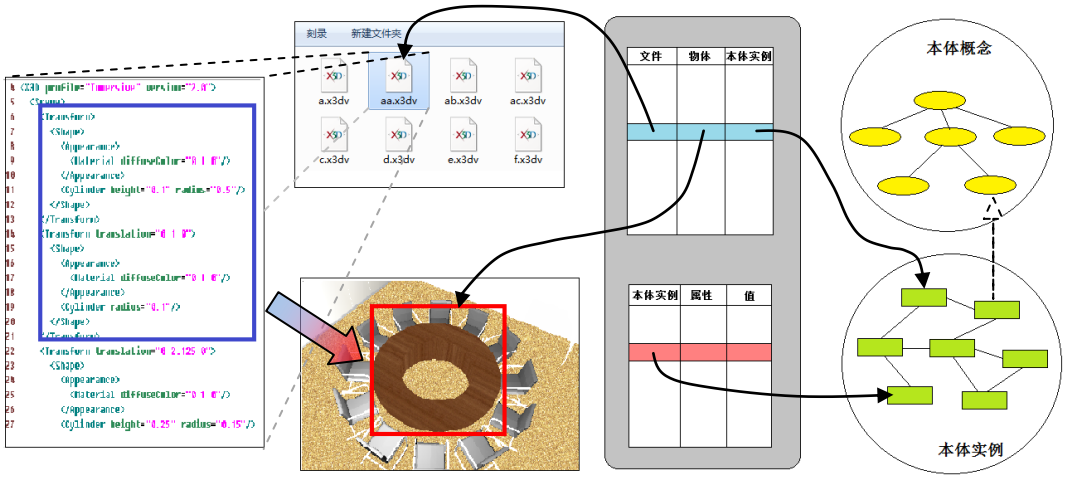

图 2-4 标注内容示意图

Fig.2-4 Demonstration of Annotation Content

图中灰色区域上方的数据表中的三列分别对应到文件系统中的某一个 X3D 文件、

该文件中的一个节点（可能包含更多子节点），也即该节点所描述的实际物体、以及该物体所对应的本体实例。灰色区域下方的数据表中的三列分别描述了一条本体知识的对象，属性和属性值，其中属性值可以是另一个本体对象或者简单数值。上下两张表通过各自的本体实例这一列进行关联。通常，同一个本体实例将出现在两张表中，分别记录它所对应的物体在文件系统中的位置，以及它所具有的属性（可存在多条）。

2.4 本章小结

本章通过实际的应用场景引出了本文所提出的面向 X3D 基于本体的半自动化语义标注系统的总体框架。本章随后利用一个典型的标注流程，简要描述了框架中各个模块的主要作用。最后，本章介绍了由标注框架产生的标注内容在持久化后的具体表现形式。后续章节将对这些模块进行详细展开，来讨论它们的内部实现细节，以及这些实现方式的利弊。

3 三维场景文档的本体描述

本章将详细介绍用本体来描述三维场景文档各方面信息的方法。我们将首先概要地介绍本文所提出的三类本体的概念以及这样做的目的和作用。随后我们将分别阐述文档本体和通用本体所需要满足的条件，并给出它们典型的实现例子。接下来本章将对用 OWL 进行本体描述中涉及到的若干要点进行讨论。本章最后将阐述自动化标注系统如何从已有的场景文档中抽取出各种可视化信息，并映射到通用本体实例的算法。

3.1 本体模型的组织结构

本文提出的用于标注的本体模型以三种类型的形式呈现，分别称为通用本体，领域本体和文档本体。其中，文档本体独立于其他两种。而通用本体和领域本体则存在扩展和利用关系，可以被看作是上下的层级关系。

3.1.1 通用本体和领域本体的基本概念

通用本体表示的是三维场景建模中最基础的概念和它们之间的关系。这些概念通常包含了单个物体的外观信息（包括视觉信息如颜色，和几何信息如形状等），以及多个物体之间的空间关系（如相邻，相交，或相离等）。这些信息是场景建模中最基础的物理信息，它们与建立它们所在的领域没有直接关系，也不带有上层领域内的任何语义信息。

领域本体则定义了一个特定领域内的概念以及它们之间的关系。例如，在家居装潢领域，桌椅可能是常见的概念，但并不是所有的场景建模领域都需要它们，因此它们不能归为通用本体中的概念。相反，更为抽象的概念，如固体物体，或者环境特征等（烟雾等），才被视为可能在大多数场景建模领域内出现的通用概念。我们将在下一节中详细阐述如何构造一个通用本体。

本文所提出的通用本体主要起到两个作用：第一，抽象出三维场景建模中的通用概念，使得它在不同领域内的语义标注过程中能够得到复用；第二，为底层的三维文档所表示的可视化信息提供一个统一的表示形式。我们在这里定义术语**可视化信息**为一个三维场景在渲染后可以被观测到的信息，包括物体的大小，形状，色彩材质等外表属性，以及多个物体间的位置关系等。在后文中，我们将频繁地用到该术语来指代从 X3D 文

档中抽取出来的场景信息。

在实际使用中，用户很可能需要在多个领域内进行半自动化标注。在这个过程中，通用本体所表示的那部分知识，正如同其名字所表示的，是重复的，通用的。因此，将这部分知识提取出来让其在各领域标注过程中复用，可以达到一次构造，各处使用的效果，从而减少一定的工作量。

我们可以根据这两种作用的不同在概念上将通用本体分成两部分，如图 3.1 所示，分别称为基本建模概念，以及可视化概念。基本建模概念主要用来被继承扩展出领域本体，因此包含那些继承层次顶端的概念，例如物体，固体物体，流体物体等，都属于此范畴。用户在特定领域内的物体概念很可能将继承于它们。另一方面，可视化概念则主要用于表示三维文档中的可视化信息，如色彩材质等。此类概念一般具有独立的意义，除特殊情况外不会被其他概念所继承。

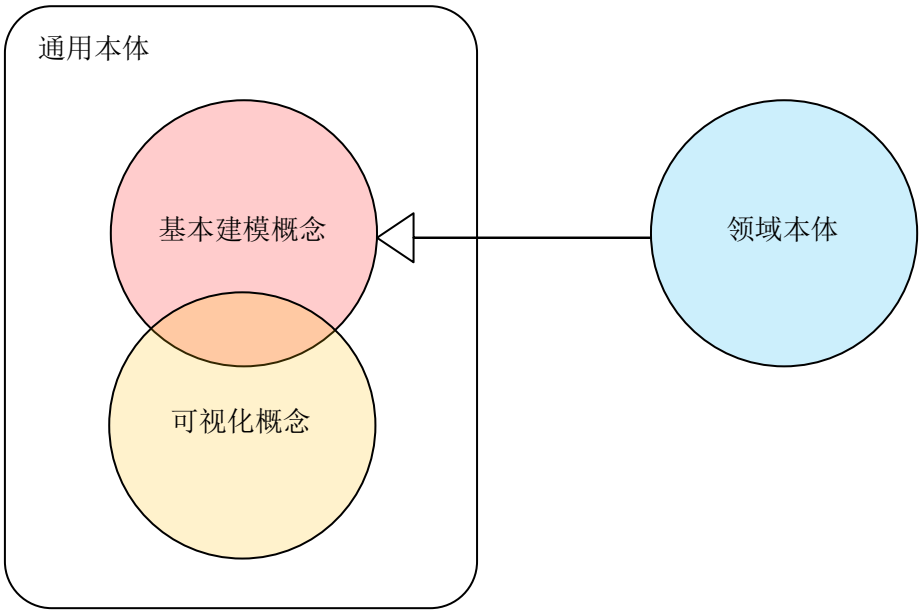

图 3-1　通用本体与领域本体的关系

Fig.3-1 Relationship between General Ontology and Domain Ontology

本文提出的标注框架中多处用到了本体作为知识的载体。话句话说，本文的框架在标注过程中产生了较多以本体来描述的中间产物。表 3-1 列举了这些中间数据会利用到的本体类型。在表中，圆形符号所占的单元格表示某种数据结构可能会用到某种类型的本体。例如用户可以在部分标注中用到通用本体，尽管通用本体所表示的可视化信息一般不作为最后的标注内容，但在特殊的需求中，用户仍然可以通过这种方式来自定义特有的标注。

表 3-1 不同中间数据利用到的本体类型

Table3-1 Types of Ontology used by Various Internal Data

本体 / 数据	通用本体	领域本体
标注规则	✓	✓
场景可视化信息	✓	
部分标注	●	✓
自动生成的标注	●	✓

3.1.2 通用本体和领域本体的相互关系

在建立了通用本体以后，我们可以利用已有的本体概念及属性来定义部分的领域本体。在这个过程中，我们通常可以采用两种方式。第一种方式是继承通用本体中的类或者属性来完成扩展。例如，假设通用本体中有一个物体类表示所有实体物体的最高抽象概念，则我们可以在其之下定义固态物体或者流体物体，从而开始我们实际应用中的物体层次关系的定义。第二种方式是利用已有的概念和属性进行新概念的定义。一个典型的例子是将一间具有指定贴图外观的房间定义为某种类型的房间。其中房间本身已经是领域本体内的类，而具有指定贴图外观则是属于通用本体范畴内的知识，因此我们可以利用通用本体进行一些最基本的概念定义。而对于更复杂的逻辑，我们则需要本文后续章节提出的标注规则来完成。

另一方面，它们之间的关系决定了它们需要以同一种表达方式来描述。在标注这层意义上，我们因为追求精确性而选择本体作为标注知识的载体，因此领域知识将用本体来描述。而可视化信息却未必如此，但是选择本体作为可视化信息的描述方式为我们带来的好处是：这两种知识具有良好的一致性。

如果我们反其道而行之，那么，我们非但无法使得领域本体直接利用通用本体来构建，在后续操作中将通用知识所表示的物体基础属性与标注规则进行匹配时，还将无法利用到本体本身携带的一部分推理功能，从而增加了规则引擎本身的工作量。图 3.2 展示了这一对选择的对比。从图中可以看出，我们后续基于规则的标注都是在本体的环境内进行，因此将可视化信息用通用本体的形式来描述是最适合的方式。如果我们另辟蹊径用别的格式来表达可视化信息，则我们需要一层额外的转换过程来达到现有的状态，

这一步是完全可以避免的。

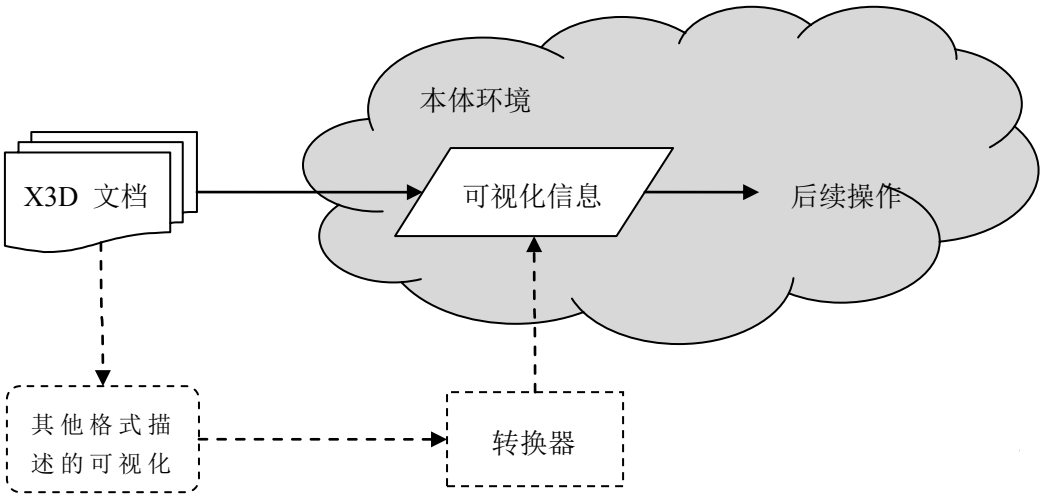

图 3.2 不同表达形式作为可视化信息的载体的对比

Fig.3-2 Comparison between Different Representations of Visual Information

3.1.3 文档本体模型

文档本体用来描述 X3D 文档的结构特征，在该本体之上创建出来的实例则被用来表述某一个特定的 X3D 文档中一部分节点间的属性及相互关系。文档本体将被用于规则的描述中，在第四章得到详细的应用。图 3-3 展示了文档本体中若干类与属性的关系。在图中我们可以看到表述 XML 文档的节点类，属性类，文档类，以及节点间相互包含或者兄弟的关系属性。

我们将这一部分本体提取出来单独成为一支所带来的好处是，之前的本体和 X3D 的具体编码格式是完全解耦的，也即之前的两层本体完全不关心一个具体的 X3D 文档在文本意义上是如何构造的。在未来的实现中，如果我们希望不再针对 X3D 文档，而采用别的三维模型文档作为标注的对象，则我们只需要替换相应的文档本体即可，而不需要修改已有的通用本体以及领域本体。

3.2 通用本体模型

3.2.1 通用本体的总体概念

本文所提出的通用本体模型并不是一个固定的模型，而是更类似于一个标准或者一

种规范。不同的通用本体模型实现可以有一些差异，但是在总体上，它们都应当满足本文对通用本体的定义以及要求。更确切地说，这些本体将能够反映三维场景建模中的基础概念。此外，由于本文的标注过程关注于 X3D 文档格式，因此我们也要求所建立的通用本体能够表达 X3D 中用户关注的那些特性。例如， X3D 文件特性包括了动画这样的元素，因此如果在一个特定的领域内所创建的场景文档使用到了这一元素，并且需要得到标注的话，则其使用的通用本体应当能够表示动画这一特性。

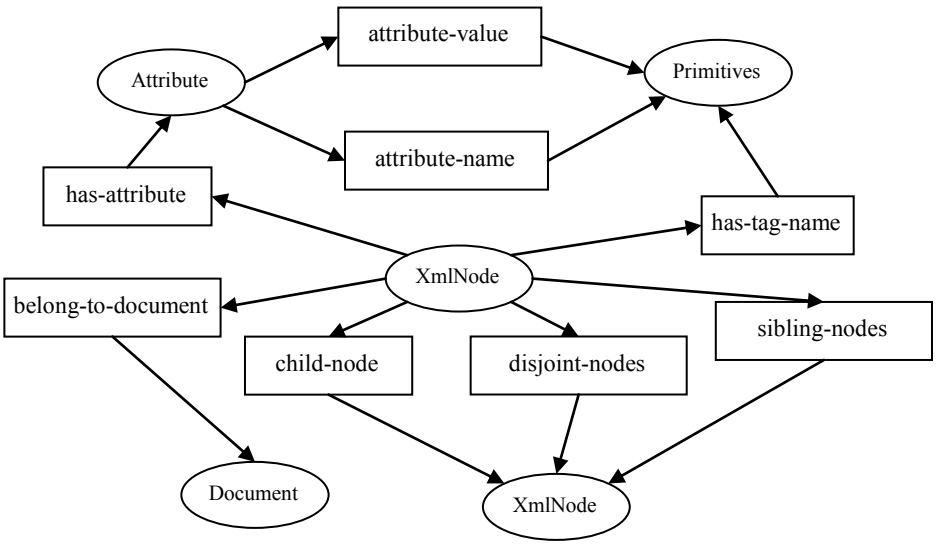

图 3-3 文档本体若干概念与属性

Fig.3-3 Several Concepts and Properties from Document Ontology

以下将提出一个可能的通用本体的实现，这个本体也将成为本文后续章节中举例或验证时使用的通用本体。在满足本文所提出的关于通用本体的需求的同时，我们力求将该本体设计得简洁而同时又不失可扩展性，以备不同的用户有不同的扩展需求。在选择概念时，我们偏向于选择那些最容易被用户用来描述一个场景的概念。

3.2.2 通用本体中的实体概念和环境概念

实体概念和环境概念在三维场景建模中表示那些可以被单独识别并提取出来的物体。其中**实体概念**描述是的那些有良好的固体形态的实际可见的物体。物体类（Object）和场景类（Scene）都属于这一概念分类。后两者的区别在于，场景类将整个场景或者世界看作一个整体，而物体类更多地是指单个存在的物体。因此场景类可以独立拥有光照以及背景信息等属性。实际标注中大多数关于物体的标注都将牵涉到该概念分类，尤其是物体类。场景类将在当整个场景作为整体被标注时被用到。

环境概念和实体概念较为相像，都表示实际存在的物体。不同的是，环境概念通常不是处于固定坐标的物体，而是遍布在更大范围（可能是整个场景，或者部分场景）内的物体。此类概念包括背景信息，灯光以及烟雾等概念。对应于上述第一类概念中所描述，只有场景类可以与环境概念的实例进行关联，因为由环境概念所表示的物体通常是覆盖大范围区域的，它们实际的渲染效果与它们在 X3D 文档层次结构中定义的位置没有直接的关系。

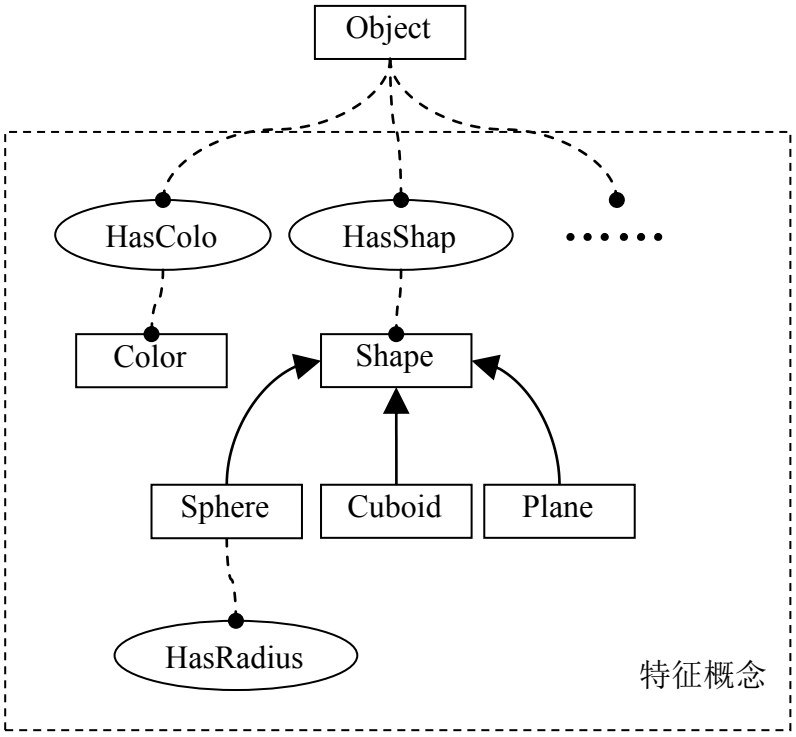

图 3-4 通用本体模型的实体概念与特征概念

Fig.3-4 Entity Concept and Feature Concept from General Ontology

3.2.3 通用本体中的特征概念

特征概念作为第二类概念被用于描述和实体概念关联的各种属性。一个特征概念的实例在和一个实体概念实例关联之前，其本身将不会有任何可视效果。例如，一个最典型的特征概念为色彩概念。一种颜色本身将不会包含任何有用信息，除非我们将其与一个实体（比如盒子）进行关联，后者将是一个实体类的实例。除了颜色以外，包括形状，材质在内诸多可视化信息属性都属于特征概念范畴。特征概念属于上一节中提到的通用本体可视化概念，因为它们更多地出现在标注规则中而非实际标注内容中。

上述介绍的实体概念以及特征概念在最后的本体描述中都将以本体类的形式存在，

为了将它们联系起来，我们在本体建模中还需要属性（Property）。显然，为了表述物体本身的特征属性，例如一个物体具有木头的材质，我们需要 HasMaterial 这样的属性来将该物体实例本身，以及材质实例关联起来。图 3-4 展示了实体概念与特征概念之间的本体关系，以及几个此类属性的示例。

3.2.4 通用本体中的行为概念

行为概念包含了用于描述物体运动方式的概念类。这一概念类主要对应于 X3D 文档中的动画元素。行为概念并不是通用本体中的必要概念，尤其在一个所要针对的场景文档完全不会用到动态物体的领域中使用时。本小节主要用它来展示通用本体收入更多建模元素的扩展能力。在本文的后续章节中，将主要针对前三种本体概念。

图 3-5 展示了通用本体行为概念的一部分。在图中可以看到旋转是行为属性的一个具体类。在一个地球围绕太阳旋转的场景中，旋转这个行为很可能会作为观察者对该场景的主要描述之一，因此在这样一个应用场景中，行为概念将得到充分使用。

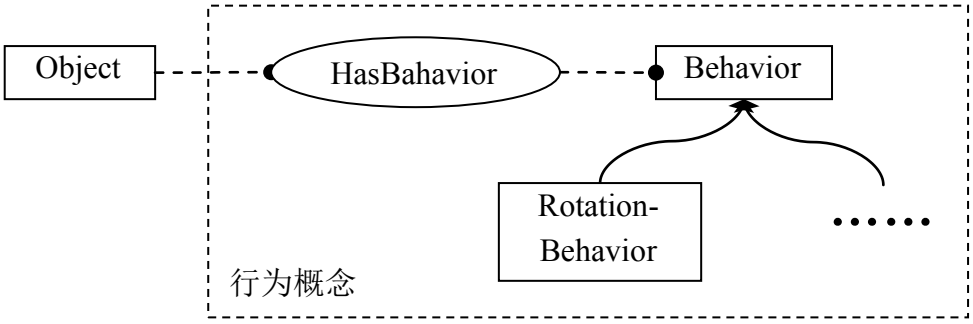

图 3-5 通用本体模型的行为概念

Fig.3-5 Behavior Concept from General Ontology

3.2.5 通用本体中的物体相互关系

为了定义物体间的相互关系，我们定义了一整个家族的属性，用来描述多个物体间的空间关系。这些属性，包括包含，相邻，分离，相交等，根据它们自身的语义信息，以及在实际表达中的形式，在我们的通用本体中可以用两种方式划分开来。

首先，从牵涉到的物体数量角度来讲，这些属性可以被划分成两个物体间的关系和多个物体与一个物体间的关系。其中大部分属性都被定义在两个物体之上，少数属性，如包围关系，则被定义在一个集合的物体和另一个单独的物体之上。另外，从属性本身的性质上将，它们又可以被分为两部分，分别称为**定性属性**以及**定量属性**。

定性属性，比如包含关系，只是直接陈述一个事实，即某个物体是否包含另一个物体。在实际的本体表达中，我们将其实现为本体的一个属性。定量属性则是一个可以被附加额外信息的属性，因此在我们的实际实现中，它们被表达成一个本体类，那些额外的附加信息则被表示成该类实例自身的本体属性。一个典型的量化属性是相离属性，它的附件信息用来表示两个相离的物体具体分开的距离。图 3-6 展示了通用本体的定性属性以及定量属性的本体关系。表 3-2 列举了若干这两类的属性、它们在几何上的含义、以及它们分别属于哪种属性，若属于定量属性又具有哪些量化的性质。

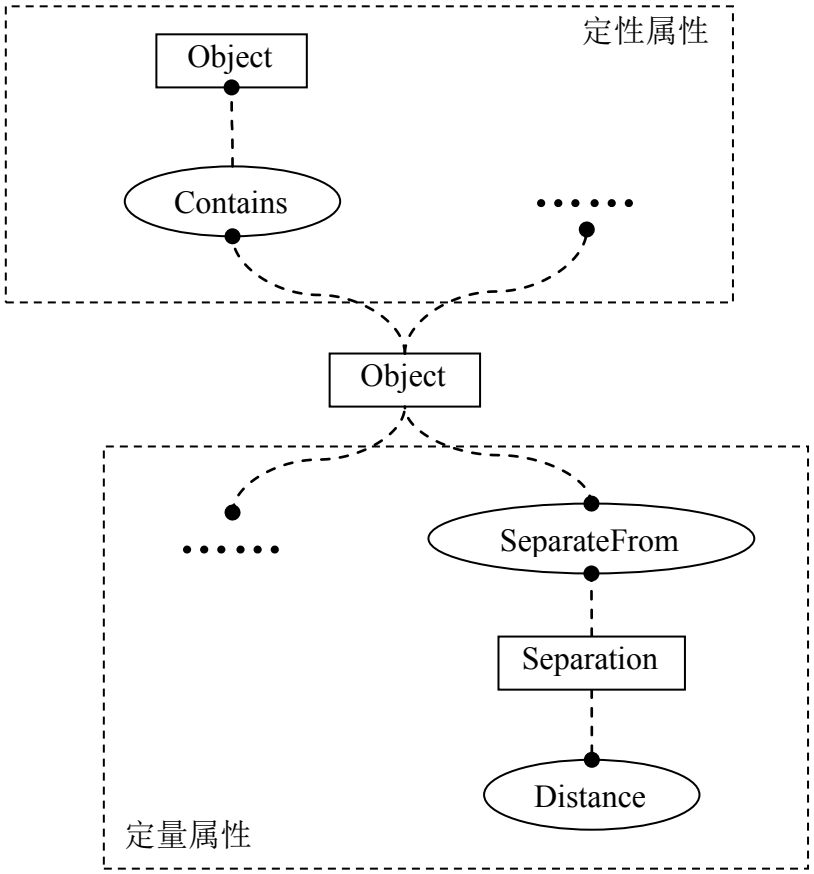

图 3-6　通用本体模型的定性与定量属性

Fig.3-6 Qualitative Property and Quantitative Property from General Ontology

在该表中，每一行都讨论了一个属性，该属性作用于两个物体 u 与 v 上，它们的语义关系见于表中第一列。表中第二列用几何语言描述了 u 与 v 需要满足的条件，其中 bb 函数代表物体的包围盒。量化属性一列中列出了某一特定属性可能拥有的量化属性值，若该栏为空则表示该属性为定性属性。另外，从该表中还可以看出，两个物体的相互关系并不是唯一的，例如 u 和 v 在存在包含关系的同时还可以存在 u 处于 v 中央的关系。

表 3-2 部分通用本体属性

Table3-2 Several Properties from General Ontology

属性	含义	量化属性	备注
u 包含 v	$bb(u) \supseteq bb(v)$	---	---
u 与 v 相邻	$bb(u) \cap bb(v) = \emptyset$ 且 $\nexists w$: w 处于 u, v 中间	---	---
u 与 v 分离	$bb(u) \cap bb(v) = \emptyset$	分离距离	---
u 与 v 相交	$\frac{bb(u) \cap bb(v)}{bb(u)} \geq f(u,v)$	相交部分占 u 的比例 相交部分占 v 的比例	f 为阈值函数，表示物体相交程度的下限
u 处于 v 中间	v 包含 u 且 $\lvert c(u) - c(v) \rvert \leq f(u,v)$	u 边缘离 v 内侧的距离	f 为阈值函数，同上；c 为计算物体中心的函数

3.3 通用本体在 OWL 中的描述

本文采用 OWL[29]作为本体的标注语言。在实际建模中，我们利用 OWL 的功能使得构建出来的本体更具有可扩展性和易修改性。

3.3.1 通用本体类在 OWL 中的描述

上一节中提到的类（例如颜色，物体等）在实际的本体描述中并非要定义成单独的一个本体类（Class）。我们可以将某个概念来定义成一个满足若干要求的本体等同类（Equivalent Class），如图 3-7 中代码所示。

在这份代码中，我们用 OWL 语言定义了色彩这样一个类。我们没有将其直接定义成一个概念类，而是利用 OWL 的 unionOf 属性将它定义为两种类别（Class Extension）的并集，一种以 RGB 模型表示，另一种以 HSV 模型表示。这种做法使得：

1. 我们可以使用不同的内部表示形式来存储同一个抽象概念；

2. 当在以后的使用中，我们决定从一种表示形式切换到另一种的时候，可以非常方便地完成这个操作。

```
<owl:Class rdf:about="#Color">
  <owl:equivalentClass>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class>
          <owl:intersectionOf rdf:parseType="Collection">
            <owl:Restriction>
              <owl:onProperty rdf:resource="#hasRComponent" />
              <owl:allValuesFrom rdf:resource="&xsd;int" />
            </owl:Restriction>
            <!-- Definitions for Components G and B, omitted -->
          </owl:intersectionOf>
        </owl:Class>
        <owl:Class>
          <owl:intersectionOf rdf:parseType="Collection">
            <owl:Restriction>
              <owl:onProperty rdf:resource="#hasHComponent" />
              <owl:allValuesFrom rdf:resource="&xsd;int" />
            </owl:Restriction>
            <!-- Definitions for Components S and V, omitted -->
          </owl:intersectionOf>
        </owl:Class>
      </owl:unionOf>
    </owl:Class>
  </owl:equivalentClass>
</owl:Class>
```

图 3-7　利用等价类进行通用本体建模

Fig.3-7 Modeling Concepts from General Ontology using Equivalent Class

3.3.2 通用本体属性在 OWL 中的描述

图 3-8 所示代码展示了通用本体定量属性的实现概要。在代码中，separateFrom 属性的值并非是另一个物体，而是一个 Separation 类的实例，后者本身则拥有 hasSeparation-Distance 属性来定义该 Separation 的具体距离。为了使得该距离有意义，我们实际上需

要将两个不同物体的实例通过 separateFrom 关联到同一个 Separation 实例上。

```
<owl:ObjectProperty rdf:ID="separateFrom">
  <rdfs:domain rdf:resource="#Object" />
  <rdfs:range  rdf:resource="#Separation" />
</owl:ObjectProperty>

<owl:Class rdf:about="#Separation">
  <owl:equivalentClass>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasSeparationDistance" />
      <owl:allValuesFrom rdf:resource="#Distance" />
    </owl:Restriction>
  </owl:equivalentClass>
</owl:Class>

<!-- Any definition for class Distance below -->
```

图 3-8 量化属性示例代码

Fig.3-8 Example Code for Quantitative Property

这些属性的定义使得我们在整个通用本体中不仅利用到了本体作为词汇分类的功能，还可以利用其推理的功能。举例来说，我们将包含属性定义为传递属性，从而使得我们不必指出每一个间接或直接的包含关系，而只需要罗列出最少量的包含关系即可获得完整的层次信息。

3.4 X3D 中节点信息到通用本体的映射

3.4.1 映射的总体算法

为了使得标注规则可以统一地对本体进行处理，我们需要将 X3D 节点中包含的基础信息转换成本体的表达形式。这一步骤也称为**可视化信息抽取**过程。在实际的 X3D 场景描述中，一个文件包含了一个 scene 节点，而一个 scene 节点又包含了诸多子节点，其中的每一个都可能表示很多物体，而每一个物体本身又可能由多个节点组成。计算机的自动处理不能像人类一样快速识别出不同物体的分隔界线，因此在这个过程，我们采取计算所有节点的属性的方式。

对于一个由一个大节点表示的物体来说，它的某些属性可能不是由该节点本身来给出，而是存放于它的子节点内部。这种情况下，我们需要有能力将子节点的属性值合并起来得到外部大节点的属性。为此我们识别出两种属性，分别称为**可合并属性**和**不可合并属性**。可合并属性表示一个节点在该属性上的值可以由它的子节点计算获得，不可合并属性则相反。大多表示物体外观的属性均为可合并属性，这其中包括色彩，大小等。不可合并属性主要包括行为属性。

算法 3.1 列出了计算一个节点内部所有属性的总体算法。该算法的输入有两个参数，其中 node 表示将要进行属性值计算的节点，all-properties 表示所有属性的集合。该算法是一个递归调用的算法，对一个节点来说，它首先计算它所有子节点的属性，然后再根据属性的可合并能力来决定自身属性的计算值。对于可合并的属性，该算法在计算完所有子节点的属性后，调用 merge 函数进行合并计算，我们将在 3.4.3 小节中介绍合并算法。最后，对于任何可合并属性，都存在一个最底层的节点，在那个节点上我们需要把它当作不可合并属性一样对待，这种情况将在 3.4.2 小节中讨论。

Algorithm 3.1: Calculate-Property
Input: node
Input: all-properties

```
1  foreach sub-node in node.children-nodes do
2     Calculate-Property (sub-node, all-properties);
3  end foreach
4  let mergeable-properties ← ⋃_{sub−node}^{node.children} sub − node. properties
5  foreach property in mergeable-properties do
6     let value ← merge(node, node.children, property);
7     node.properties[property] ← value;
8  end foreach
9  foreach property in {all-properties − mergeable-properties} do
10    let value ← Calculate-Single-Property-Value(node, property);
11    node.properties[property] ← value;
12 end foreach
```

3.4.2 单个节点通用本体概念的抽取

（1）实体概念和环境概念的抽取

对于实体概念，我们在 X3D 文档中寻找那些用于构造物体形状的节点。这些节点包括基础形状（cube,sphere）等节点，以及高级样式的节点（如 NurbsCurve）这些节点

将作为最底层的物体节点被一一映射成通用本体中的物体类。

对于环境概念，我们在 X3D 场景中寻找特定名称的节点，例如 fog 等，我们把这些节点根据节点标签名字映射成相应的本体概念。此外，对于任意一个 X3D 文档，我们将其整个 scene 节点映射成一个场景概念的实例。

对于任何一个最原子的物体，它的外部包含了 Appearance 等节点，我们在做实体概念的抽取时直接将它们计做一个节点，从而省却了之后多层节点都需要计算属性值的工作。这一步骤称为节点的扁平化。图 3-9 展示了它的主要思想。

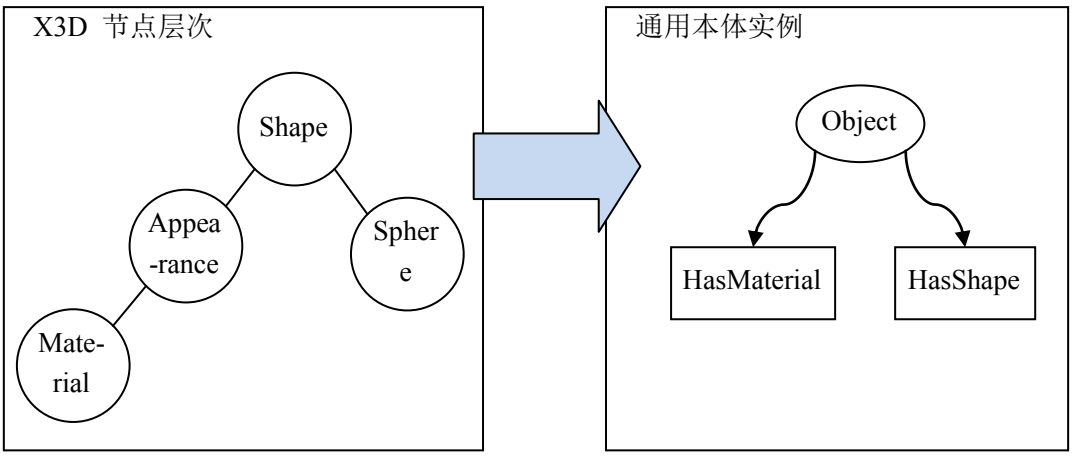

图 3-9 形状节点的扁平化

Fig.3-9 Flattening of Shape Node

（2）特征概念的抽取

X3D 虽然拥有着 XML 良好层次结构的特点，但是同一类信息在它的表示中却并不单纯地处于固定的位置，而是可能散落在不同的场所[30,31]。例如，X3D 格式虽然允许我们在一个物体的 Appearance 节点下的 Material 下定义一个 color 属性，然而一个物体的实际渲染色彩却未必由该值决定，它很可能由物体表面贴的纹理图片所决定。因此在一个 X3D 文档中获取相应的外观信息并不是一个简单的对 XML 做机械化检索的过程，而是需要一些智能的处理。

具体来说，当尝试转换一个特定的属性时，我们将搜索相关节点的上下文来搜集有关信息，并通过计算获得最终的属性值，随后用该属性值来构建本体实例，以及实例相关联的属性信息。以色彩属性为例，给定一个物体节点，我们将需要获取该物体几何形状自身定义的色彩，以及分析它的贴图色彩，来决定是否能计算出一个主要色彩。该过程的结果将作为物体的色彩属性值，用来构造色彩本体实例，并和已知的物体所对应的

本体实例进行关联。对于不同的属性，我们所需要搜索的上下文和计算方式都是不同的。我们将在第五章来阐述如何配置这些信息，使得自动标注引擎可以根据所要获取信息种类的不同，应用不同的算法。

（3）物体间相互关系的抽取

对于任意的两个已经抽取出的物体，我们记录着他们所对应的 X3D 文档中的节点位置。在这个基础上，我们可以依照着表 3-2 中提出的定义对它们进行空间关系属性的判断。对于判断为真的物体，我们为它们创建相应的通用本体属性关联。在这个过程中，判断条件是存在一定人为主观因素的，因此其计算公式本身也应存在修改的可能。我们将在第五章系统实现中来讨论如何使得这些公式的计算成为可配置的项目。

表 3-3 部分属性及判断公式

Table3-3 Several Properties and Their Decision Formulae

属性	判断公式
u 与 v 相交	$\bigwedge_{i=x,y,z}\bigwedge_{\omega=(0,0,0)}^{(2\pi,2\pi,2\pi)}[\max_i^{\omega}(u) > \min_i^{\omega}(v) \vee \max_i^{\omega}(v) > \min_i^{\omega}(u)] \bigwedge$ $\bigwedge_{i=x,y,z}\bigwedge_{\omega=(0,0,0)}^{(2\pi,2\pi,2\pi)}[\min\{\lvert\max_i^{\omega}(u)-\min_i^{\omega}(v)\rvert,\lvert\max_i^{\omega}(v)-\min_i^{\omega}(u)\rvert\} > f_{uv} * \max\{\lvert\max_i^{\omega}(u)-\min_i^{\omega}(v)\rvert,\lvert\max_i^{\omega}(v)-\min_i^{\omega}(u)\rvert\}]$
u 与 v 相离	$\bigvee_{\omega=(0,0,0)}^{(2\pi,2\pi,2\pi)}\bigwedge_{i=x,y,z}[\max_i^{\omega}(u) > \min_i^{\omega}(v) \vee \max_i^{\omega}(v) > \min_i^{\omega}(u)]$
u 包含 v	$\bigwedge_{i=x,y,z}\bigwedge_{\omega=(0,0,0)}^{(2\pi,2\pi,2\pi)}[\max_i^{\omega}(u) > \max_i^{\omega}(v) \vee \min_i^{\omega}(v) > \min_i^{\omega}(u)]$

表 3-3 列出了部分属性的计算公式。在这些公式中，我们引入以下两个函数：

1. $\max_i^{\omega}(u)$ 计算物体 u 在经过ω旋转后，在 i 坐标轴上的最大坐标

2. $\min_i^{\omega}(u)$ 计算物体 u 在经过ω旋转后，在 i 坐标轴上的最小坐标

其中 i 坐标轴可以是三维空间内的任意三个坐标轴之一。

该表中列出了三个基本属性的判断方式。其余属性可以由类似的计算，或者利用这

些基本属性加上额外的计算来获得。

3.4.3 本体属性值的合并

在将节点内属性转向本体实例的过程中另一个需要处理的问题是属性的合并，即两个小物体合并组成了一个大的物体，则如何根据小物体自身的属性值来确定合并后大物体的属性值。若给定节点 x 以及它的子节点集合 sub-nodes，并假设后者的属性值已经计算完毕，则对于任意一个可合并属性 p，我们需要确定函数 combine 来计算属性 p 在节点 x 上的值：

$$x[p] = \text{combine}(\{(s, \text{sub-property}) \mid \text{sub-property} = s[p] \text{ and } s \in \text{sub-nodes}\})$$

在这里，我们识别出以下三种不同种类的属性合并方式：

1. 继承，即合并后物体的属性继承合并前物体的属性。具体来讲，若：

$$\{sp \mid sp = s[p] \text{ and } s \in \text{sub-nodes}\} = \{\text{sub-nodes}[0][p]\}$$ ，则：

$$x[p] = \text{sub-nodes}[0][p]$$

其中，sub-nodes[0]表示 sub-nodes 集合的任意一个元素。该种合并方式的一个可能的例子是，一些同样具有木质材质的物体合并构成了更大规模的物体，则后者可以沿用其内部物体的材质属性。

2. 舍弃，即完全不考虑内部物体的属性。这种情况下，x 在 p 上的属性由单个节点的计算公式决定：

$$x[p] = \text{Calculate-Single-Property-Value}(x, p)$$

行为概念的合并通常用到此种方式。考虑一个恒星-行星-卫星系统内部的旋转属性。对于模型的构建者和观察者来说，只有外部的旋转和内部的旋转是有意义的，这两种旋转的合并虽有数学意义，却很少有实际意义。事实上从通用角度讲，任何多种行为的合并几乎都涉及复杂的数学计算而变得不可预测，因此对于该类属性，通常选择舍弃属性值的合并。

3. 一般公式计算，这种方式可以引入任意复杂的计算。其最简单的形式可以是几个布尔值的或操作（表示内部物体中只要有一个物体有某属性，则外部物体也有该属性）。较复杂的形式则可能和主要色彩的计算一样，会涉及到数值的统计计算。以 size 属性为例，它的合并方式如下公式所示：

$$x[p] = \begin{bmatrix} \min_x & \min_y & \min_z \\ \max_x & \max_y & \max_z \\ s_x & s_y & s_z \end{bmatrix}$$

$$\text{其中，} \begin{cases} s_i = \max_i - \min_i,\ i \in \{x, y, z\} \\ \min_i = \min[\{m \mid m = s[p].\min_i \text{ and } s \in sub-nodes\}],\ i \in \{x, y, z\} \\ \max_i = \max[\{m \mid m = s[p].\max_i \text{ and } s \in sub-nodes\}],\ i \in \{x, y, z\} \end{cases}$$

3.5 本章小结

本章首先提出了通用本体，文档本体和领域本体这三种从不同侧面对三维场景文档进行描述的本体。在这之后，通用本体应该满足的条件以及它的一个典型实现得到了详细的阐述。本章随后讨论了用 OWL 语言来表示上述本体的几个要点。最后，本章描述了如何从 X3D 场景文档中抽取出可视化信息，并表示为通用本体知识的算法。

4 基于规则的语义标注研究

本章将详细介绍如何利用规则的定义和实施来抽象语义标注的过程。首先，本章将给出本文所提出方法中用到的标注规则的表达方式，以及在该表达下可以表述哪些逻辑。随后，本章将构建一种数据结构来表示用户定义的规则集合，继而利用它来实现一个算法，用以解决在大量场景文档中匹配规则的问题。最后，本章将讨论如何验证用户定义的规则的一致性。

4.1 基于规则的标注自动化的基本思想

基于规则的系统通常利用到它的两个特性，形式化所提供的严谨性和抽象程度所带来的潜在的自动性。在本文的研究中，我们将同时用到这两种特性，并且侧重于后者。在实际的手动标注过程中，我们发现许多的标注是类似的，我们可以抽象出它们的共性，并且让计算机系统来自动判断文档中哪些内容是符合条件的，然后对其进行自动标注。

例如，在一个办公楼的场景建模中，存在着办公室，会议室，会议桌等概念。在标注过程中，一个特定的操作是：将一间房间标注为会议室，然后将房间中间的圆桌标注为会议桌。这种重复出现的标注操作可以被看成一种标固定模式，从而得到自动化。我们可以应用**向上扩展**的标注方式，也可以应用**向下扩展**的标注方式。前者要求用户首先标注出一个会议桌，并定义规则表明在一个会议桌外围的拥有房间形状的物体将被定义为会议室；后者则需要首先标注出一个会议室，随后定义规则说明一个会议室内部中央地区的物体为会议桌。可见，在无论哪种标注方式中，用户都需要抽象出会议室与会议桌对应于标注的空间关系，并且对其中一方进行显式的标注，剩余的标注将由自动标注引擎完成。我们将这类必须完成以启动自动标注流程的标注内容称之为**部分标注**。

另一方面，这种既可向上扩展又可向下扩展的标注方式对应于三维场景本身的特性。具体来说，物体往往由更小的物体或部分组成，而场景又有多个物体组成。在不同细致层次上的物体往往因为这种组合关系而具有语义的相关性。本文设计的标注方法就是要使得这种相关性最大程度地被标注人员利用到标注规则中。

4.1.1 自动标注的基本流程

图 4-1 展示了自动标注的基本流程。该流程包含了一个不断寻找下一个可触发规则，

并不断应用它们的循环，直到没有更多这样的规则。由规则的应用所产生的新的知识一方面将作为标注内容被存储起来，另一方面它们还将反馈到规则的匹配过程中，用来触发更多的规则。一个规则成为**可触发的规则**当且仅当它所描述的模式在场景文档中存在匹配它的节点或物体，我们在某一时间点上可以应用该规则产生新的标注。

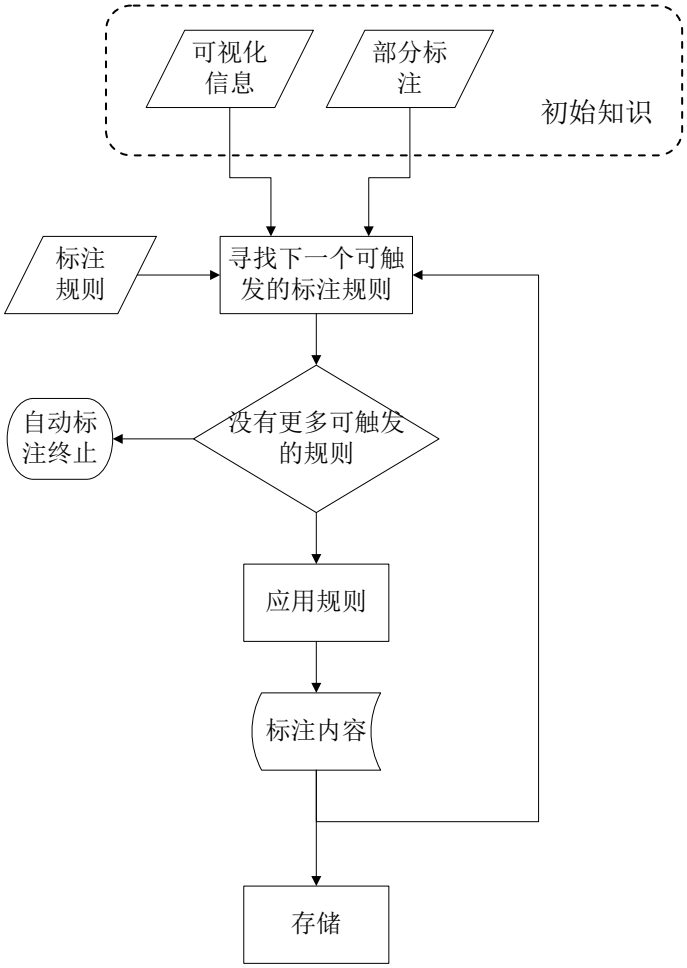

图 4-1 半自动标注基本流程

Fig.4-1 Basic Process of Semi-Automatic Annotation

在图 4-1 中，上方虚线部分所围起来的区域所包含的内容称之为标注的**初始知识**。它包含两方面，分别为由场景文档中抽取出来的可视化信息（已在第三章介绍）和由标注者完成的部分标注。这两种信息表示了标注过程开始时标注系统所拥有的所有知识。标注过程将从这些知识开始，通过规则匹配发现更多的知识，并将它们根据 2.3 小节中提出的存储方案存储为标注内容。

4.1.2 标注引擎和本体推理的关系

在利用本体进行语义标注的方法中，一种可做的选择是，利用本体本身的推理功能进行知识的扩展。例如在上面的例子中，我们可以将会议室的本体概念定义为满足某些条件的概念的等价类。但是在实例的操作中，尤其是实例属性的操作中，本体所能做的推理则较为有限。在上述例子中，假设会议室和会议桌可以用于大中小三种规模，且有如下的标注逻辑：一个特定会议室与它内部的会议桌的规模保持一致。这时，由于必须有一个实例将在自动标注的过程中产生，因此这个新产生的实例的属性就需要特殊处理。而这个推理的过程无法在本体内部完成，因而需要额外的规则引擎来进行处理。

另一方面，我们的半自动化标注引擎并没有完全抛弃那本体的推理功能而是最大化地利用了它。一个体现出该特点的例子是第三章所提到的通用本体中的包含属性。包含属性在通用本体中被定义成具有传递性质的属性，因此当物体 A 包含物体 B，而物体 B 包含物体 C 时，我们就自动拥有了物体 A 包含物体 C 这一个事实。这层推理将由本体推理机来完成。否则，我们必须手动实现描述逻辑的推理过程，这将是复杂且艰巨的任务。其推理的结果（A 包含 C）将和其他直接的包含关系一样，作为已知知识的一部分，作为规则在应用前进行模式匹配操作的依据。

在上述例子中，我们还可以发现，“空间关系”所携带的逻辑性质和本体的描述能力有着一定的重叠部分。因此，更多的空间关系可以由本体推理来发现，而标注者则需要将注意力集中到空间关系和领域关系的对应上。

4.2 规则的表示形式

4.2.1 规则的语法

本文中用到的规则将采用统一的表达形式，即用括号分割的列表形式。我们首先作以下两个定义：

定义 4.1：任何由字符组成的不包含空格或括号的字符串，称之为**原子对象**。

例如 hasColor, color, 1, chair-1 都是原子对象

定义 4.2：**列表对象**：若$x_1, x_2, \cdots\cdots, x_n$ $(n \geq 0)$ 都是原子对象或列表对象，则：$(x_1\ x_2\ \cdots\cdots\ x_n)$也是**列表对象**。

从上述定义中可以看到，用于构造列表对象的基础结构只有一种（即任意字符串），而组合这些对象的方法也只有一种，因此这种结构具有语法简明，易于处理的优点。另

外，由于列表对象的定义是递归的，使得我们可以创造复杂的列表对象，因而它同时又具有表达能力强的优势。在此基础上，我们可以定义本文所提出的标注规则：

定义 4.3: 一个**标注规则**是一个合法的二元列表对象，以如下所示的形式表示。

(condition action)

定义 4.4: 在一个合法的标注规则中，我们称它的两个元素分别为规则的**条件部分**和**行为部分**。它们共同组成一个规则来表示当某些条件满足时，执行某些特定的行为。在本文中，这些行为特指要生成的标注内容。

4.2.2 标注规则中的合法标识符

图 4-2 列出了标注规则中可用的合法标识符以及它们间的相互关系，以下我们将对它们进行一一介绍。

图 4-2 标注规则中的合法标识符

Fig.4-2 Valid Identifiers in Annotation Rules

1. 简单标识符

简单标识符包括变量与关键字两种。变量名用于在规则内部引入一个命名对象，在规则的其余部分可以对该对象做属性的约束。任何原子对象可作为变量名。关键字包括 and, or 和 not 三种逻辑运算符，它们用于构建复杂的规则条件，将在后文中定义。

2. 本体标识符

本体标识符包括任何本体类的名字以及本体属性的名字。本体的范围包括本文第三章介绍的通用本体，领域本体、文档本体、以及 OWL 本身定义的各种基础本体类，如 Thing, Class 等。

3. 基本函数标识符

基本函数标识符表示作用基本类型（如数字，字符串等）的函数。这些函数包括两种，一种用于结合多个基本类型并产生新的基本类型对象，我们将其称之为基本操作符。它们包括数值对象的基本运算，以及字符串对象的拼接等操作。另一种函数作用于单个基本类型对象并返回布尔值（真或假），我们将它们称之为基本属性。若干基本属性在表 4-1 中被列出。

表 4-1 部分基本属性

Table4-1 Several Basic Properties

类别	属性名	描述
数值相关	greater-than	(greater-than x y)表示数值上 x 大于 y
	numeral-equals	(numeral-equals x y)表示数值上 x 等于 y
字符串相关	string-equals	(string-equals x y)表示 x 与 y 表示的字符串相等
	string-prefix	(string-prefix x y)表示作为字符串 x 是 y 的前缀子串

4. 系统操作符标识符

系统操作符包括本体操作符和文档操作符。它们分别用于获取某个本体实例的相关属性，或者某个文档内容的相关属性。在本文中，文档即以 XML 形式存在的 X3D 文档，因此文档属性操作的对象是 XML 节点。如果在后续工作中，我们希望更换文档的类型，则该规则的标识符中我们只需替换文档本体，以及文档操作符（图 4-2 中用阴影部分圈起来的部分）即可。表 4-2 列出了部分本体操作符以及文档操作符。

4.2.3 标注规则的条件部分

以下，我们将详细定义一个合法的规则的条件部分的构成。

定义 4.5：原子条件：原子条件是最基本的条件，它由如下形式的三元组组成：

(predicate object1 object2)

它所表达的意思是实体 object1 与 object2 之间存在着由谓词 predicate 所表达的关系。判断 object1 和 object2 是否具有 predicate 所表示的关系需要我们的标注系统知道 predicate 的语义，因此我们要求谓词 predicate 必须是已知本体中的一个属性或者系统预定义的属性之一。例如之前例子中的“x 是一个 XmlNode”这个条件将被表示成：

(instance-of x XmlNode)

表 4-2 部分系统操作符举例

Table4-2 Examples of System Operators

操作符名称	类型	参数	参数类型	作用
instance-of	本体	node	本体实例	(instance-of x y) 表示 x 是 y 本体类的一个实例，也可以是 y 本体类子类的一个实例
strict-instance-of	本体	node	本体实例	(strict-instance-of x y)表示 x 是 y 本体类的一个实例，但不是 y 本体类子类的实例
get-ontology-instance	文档	node	Xml 节点	返回 node 节点所对应的本体实例
get-node-attribute-value	文档	node	Xml 节点	返回 node 节点的 attribute 属性的值
		attribute	字符串	
get-node	本体	ont-inst	本体实例	返回 ont-inst 实例所对应的 xml 节点
get-property-value	本体	ont-inst	本体实例	返回 ont-inst 实例的 property 属性值（可能为集合）
		property	字符串	
get-unique-property-value	本体	ont-inst	本体实例	返回 ont-inst 实例的 property 属性值的唯一值
		property	字符串	

这其中 instance-of 就是一个谓词，它来自系统预定义的本体操作符，表示本体实例和本体类的属于关系，x 则表示一个变量，而 XmlNode 作为来自于文档本体中已定义

的本体类。

在定义了基本的原子条件之后，我们可以用逻辑操作对它们进行组合，来完成更复杂的条件构造。严格来说，有如下定义：

定义 4.6: 复合条件：若 $x_1, x_2, \cdots\cdots, x_n$ $(n \geq 0)$ 都是合法的原子条件或者复合条件，则：(**and** x_1 x_2 …… x_n) 以及 (**or** x_1 x_2 …… x_n) 也是合法的条件。如果 x 是合法的原子条件或复合条件，则(**not** x) 也是合法的条件。上述三种条件称之为复合条件。

从上述定义中可以看出，and 和 or 的参数都不受限制，因此由它们引导的条件将不再是三元组，而是可以任意的 n 元组。为了和这两种表达式的语法语义保持一致，我们选择在原子条件中将谓词放在所有元素的最前，如(instance-of x XmlNode) 而非更加直观的(x instance-of XmlNode)。下文将紧接着给出该种复合条件的一个例子。

4.2.4 标注规则的行为部分

规则主体部分的后一半行为部分表示当前半段的条件部分满足时，将做出哪些具体标注。关于标注行为的表示，我们有如下定义：

定义 4.7. 原子标注行为：原子标注行为是最基本的标注行为，它表示标注行为中的单个标注。它在规则中的描述和定义和原子条件一样，也采用三元组的形式。这些标注行为主要以两种形式存在：

（1）将某一个 X3D 的节点对应到一个本体实例

（2）为某个本体实例的属性进行赋值

例如对于上述第一种用法，我们使用如下的语句：

(instance-of x SomeDomainOntology)

可以看到，这个表达方式和在条件部分中使用的是一致的。所不同的是，当该表达式出现在条件部分中时，我们用它来申明：存在这样一个变量 x，满足某个条件。而当该表达式出现在行为部分中时，我们用它来创造一个 x 变量，它满足该条件。在这里，我们实际上创造了一个变量 x，使它成为 SomeDomainOntology 本体类的一个实例。

对于上述第二种用法，我们使用如下语句：

(somePropertyOfDomainOntology x y)

同样地，该语法和条件部分一致，只不过在这里，我们显示地将 x 和 y 用本体属性关联了起来。我们将在标注行为的定义后给出它们的实例。

定义 4.8. 标注行为：标注行为是将多个原子标注行为绑在一起形成的列表对象。严格地说，

若 $x_1, x_2, \cdots\cdots, x_n\ (n \geq 0)$ 都是合法的原子标注行为，则：

$(x_1\ x_2\ \cdots\cdots\ x_n)$ 是合法的标注行为。

如以下完整的一个标注行为表示了两个操作：首先建立一个 MeetingRoom 的本体实例记为 x，随后将 x 的 hasCapacity 属性设置为 10.

((instance-of x MeetingRoom) (hasCapacity x 10))

对于这个例子，如果我们之前已经为 X3D 中的某个节点 y 关联了本体实例，则我们还可以用系统预定义的本体操作符来简便地进行规则编写如下：

(hasCapacity (get-ontology-instance y) 10)

4.3 规则的模式匹配与应用

在第二章中的典型标注流程中我们提到，当用户指定标注规则完毕以后，将会首先经过一个规则一致性检测的阶段。在本研究中，规则的匹配及应用涉及到一张网络图的建立，而一致性检测机制也正是基于同样的图上进行，因此我们将在这一小节中首先假设用户制定的规则都是内部相互一致的，并在此基础上讨论进行规则条件的匹配。在下一小节我们将描述如何验证规则的一致性。

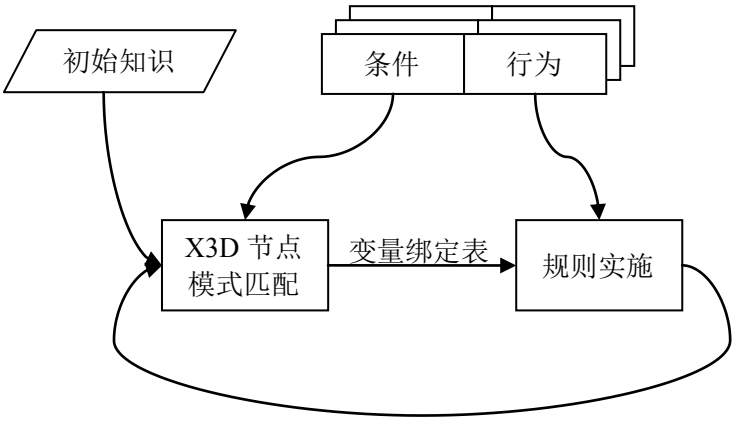

图 4-3 规则匹配与应用流程

Fig.4-3 Process of Rules Matching and Application

对于一组给定的 X3D 文档和标注规则，我们必须有能力从中找到符合标注规则条件部分的 X3D 节点或者其对应的本体实例，从而决定在标注的每一步应用哪一个规则。

该过程如图 4-3 所示，我们有两个模块，模式匹配与规则实施。它们分别从规则中读入条件部分和行为部分。模式匹配模块从知识库中寻找符合条件的规则，并将相应的变量绑定列表传送给规则实施模块，后者根据规则的行为部分和变量的绑定生成新的知识，并反馈给模式匹配模块进行更多可触发规则的搜索。

4.3.1 规则条件匹配的基本思想

本文中使用的匹配算法基于研究[32]，并根据本文的领域需求做了一定的修改而调整。我们将规则的条件部分表示成一棵树，从树的根节点到叶子节点的一条路径表示一个完整的规则条件部分。在不同规则中用到的相同的条件可以被存储在同一个节点中，从而达到判断逻辑复用，且节省空间的作用。

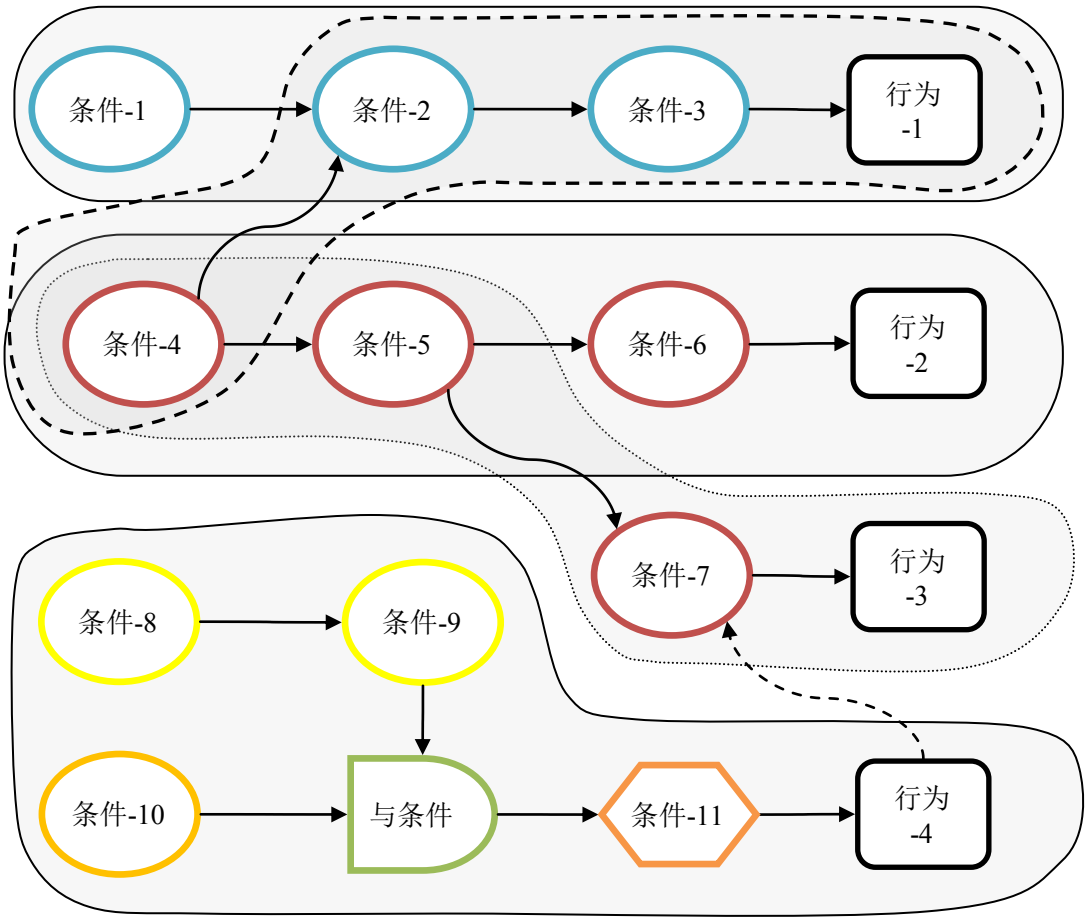

图 4-4 规则条件网络示例

Fig.4-4 Examples of Condition Net

本研究中使用的规则网络如图 4-4 所示，在图中我们可以看到存在四类节点，分别为：

1. 原子条件节点（椭圆形图元）。每个条件节点代表上一节中提到的原子条件，在该节点内存放了它所表示的条件的相关信息，包括条件使用到的谓词以及条件的变量和值。所有条件节点都最多只有一个输入，但可以有多个输出。

2. 行为节点（矩形图元）。行为节点代表标注规则中的标注行为。由于在同一个规则中，不同标注行为的地位可以认为是平等的，因此在这个示意图中，一个行为节点可以代表单一的标注行为，也可以代表一组行为的集合。行为节点可以有多个输入，也可以有多个输出。

3. 复合输入节点（半边圆弧的矩形图元）。这类节点用来表示标注规则中的复合规则，并且它们自身也可以带有条件，这些条件通常涉及到多个变量。例如上图中的“与条件”节点接受两个条件节点作为输入。复合输入节点都具有两个或两个以上的输入节点。它们和条件节点一样，同样可以有多个输出。

4. 复合条件节点（六边形图元）。这类节点表示涉及到多个变量的条件节点，它们通常出现在复合输入节点之后。

如果我们将该图看作一个有向图，则每一条从叶子条件节点开始，到行为节点的路径都对应了一个完整的标注规则。在图 4-4 中一共用不同的虚实线圈标出了 5 个这样的路径。这种路径分为两种：

定义 4.9. 简单路径：简单路径是不牵涉到复合节点的路径，如路径“条件 1→条件 2→条件 3→行为 1”，就是一条简单路径。在这条路径上的所有节点都将约束同一个物体（或同一个节点，同一个本体实例）。这些约束条件之间的关系是与的关系，即一个物体（或者节点，实例）必须满足所有的条件才能执行最后的行为节点。

定义 4.10. 复杂路径：是牵涉到复合节点的路径，如路径“（条件 8→条件 9）&条件 10→行为 4”，就是一条复杂路径。复杂路径可以看作是多条路径通过复合节点的产物，它的语义取决于所涉及到的复合节点类型。

对于“与条件”节点来说，不同输入的简单路径约束在不同的变量上，与条件节点将它们汇总起来，并增加定义自己的条件判断。例如以下的规则：

```
((and (instanceof x DomainObject)
      (instanceof y Color) (hasColor x y))
      some-action)
```

会被转换成如图 4-5 所示的网络。在这个图中，三个条件节点所引申出来的矩形框中描述了它们各自所记录的条件信息。我们在两个原子条件节点中故意省去了变量名字，

来表达这样一个思想：简单路径上的每个节点都作用于一个变量上，因此可以忽略变量的命名。然而对于复合节点来说，它就必须记录变量的名字，以及每个输入节点对应的变量名字，因为存储在它之中的条件往往涉及到多个变量。

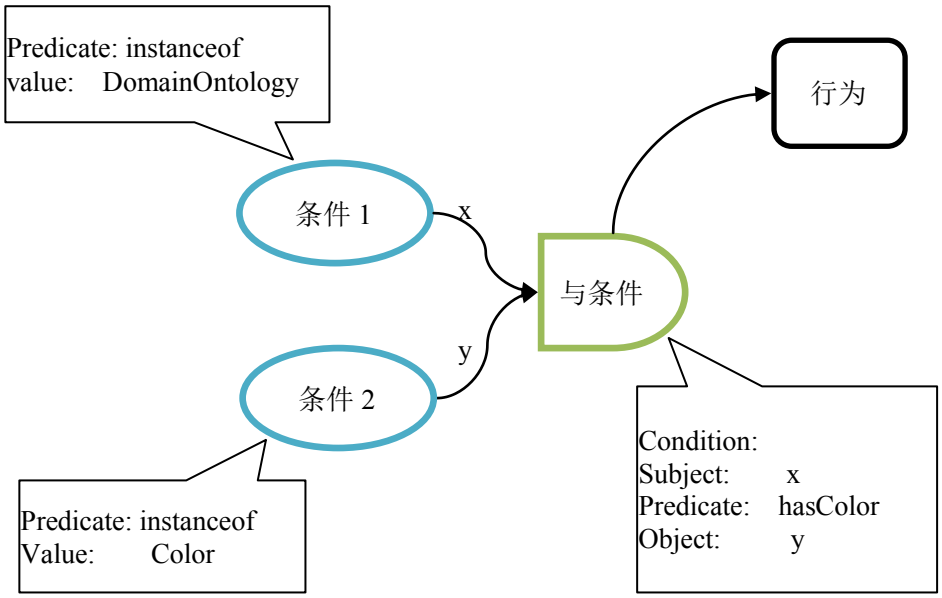

图 4-5 与条件网络图

Fig.4-5 Condition Net for And Condition

对于“或条件”节点来说，不同输入的简单路径仍然约束在同一个变量上。或节点将它们汇总起来，并不作任何额外的条件判断。这是因为，如果需要做额外的条件，那么我们只需为该或节点增加一个输入节点即可。

在图 4-4 中，另外有两种情况需要特别对待，一种是多入度的条件节点。该类节点在规则中存在被重用的情况。例如图中的条件 2 就同时被条件 1 和条件 4 指向，在实际的规则描述中，将会有两条规则同时提到了条件 2。在构造条件网络的时候我们可以将它抽象成一个单独节点，而不是为它们各自创建不同的节点。这样做的好处是：第一我们可以节省创建节点所需要的资源空间，减小网络的规模；第二可以将相同的条件判断集中到一起，在实际的规则匹配和应用中减少相同判断的次数。

另一种需要特别对待的情况是由“行为”节点出去的边。“行为”节点已经是一次单独标注的最终操作，然而由于规则行为的应用而产生的副作用很可能触发其他条件节点。因此在我们的网络图中，如果一个行为节点 a 指向一个条件节点 c，则表示实施行为 a 可能会导致 c 条件的判断结果改变。在实际标注过程中，我们可以通过这条边迅速地查找到下一次可能会被触发的条件节点，从而提高了规则匹配的效率。

4.3.2 规则匹配及应用

当我们构建完毕 4.3.1 节中提到的规则条件网络后，就可以在它之上进行规则的匹配和应用操作了。我们让本体实例暂时存储在它所符合的原子条件节点上，若它们可以满足更多的原子条件，则可以沿着网络图中的有向边流动到下一个节点中。一旦一个本体实例到达了最终的行为节点，就意味着该行为节点可以被触发，也即该规则的行为部分可以得到实施。以下我们详细地定义每个节点中包含的属性，并给出规则匹配的算法。

1. 原子条件节点的结构为：〈P, V, Neg, VI, PI〉

其中 P 表示谓词(Predicate)，V 表示值(Value)分别描述了该节点所表示的条件判断内容，Neg 表示该判断条件是否需要取反。这三个变量联合起来表示了某个变量代表的实例通过谓词 P 必须满足（或者不满足）值 V. **满足条件实例** VI(Valid-Instances) 是一个本体实例的集合，其中的元素满足从叶子节点到当前节点的路径上所有的条件。**潜在满足条件实例** PI(Potential-Instances) 同样是一个本体实例集合，其中的元素仅仅满足当前节点所表示的条件，但未必满足当前节点来源路径上的条件。

以图 4-4 中的条件-2 为例，存储在该节点满足条件实例集合中的元素同时满足条件-1 和条件-2（或者同时满足条件-4 和条件-2），而存储在该节点潜在满足条件实例集合中的元素可能只满足条件 2.

2. 行为节点的结构为：〈Actions〉

Actions 是行为节点中唯一存储的内容，它表示所要执行的行为的集合，对于其中的每一个行为，其结构为：〈N, P, V〉

其中名字 N(Name)为将要使用到或新创建的变量名，属性 P(Property)和值(Value)分别表示该变量所代表的实例将要被关联的属性名和值。

3. 与条件节点的结构为：〈I2V, VI〉

其中**输入映射** I2V(Input2Var)是一个键值对结构，它将输入节点映射到变量名上。与条件节点的后继节点都会处理多个变量，因此需要记录输入节点和变量名的一一对应关系。**满足条件实例** VI(Valid-Instances) 是另一个键值对结构，它将变量名映射到满足条件的实例的集合上。

4. 或条件节点的结构为：〈VI〉

满足条件实例 VI(Valid-Instances) 用于保存满足条件的实例的中间结果。

5. 复合条件节点的结构为：〈SV, P, OV, VI〉

它的结构与原子条件类似，区别在于它作用于两个变量之上。并且，复合条件节点

将不再包含潜在满足条件的节点。它的前三个元素分别表示牵涉到条件的两个对象SV(SubjectVar)，OV(ObjectVar)和它们之间需要满足的谓词关系 P(Predicate). 最后的 VI与前面所述的**满足条件实例**集合一致。

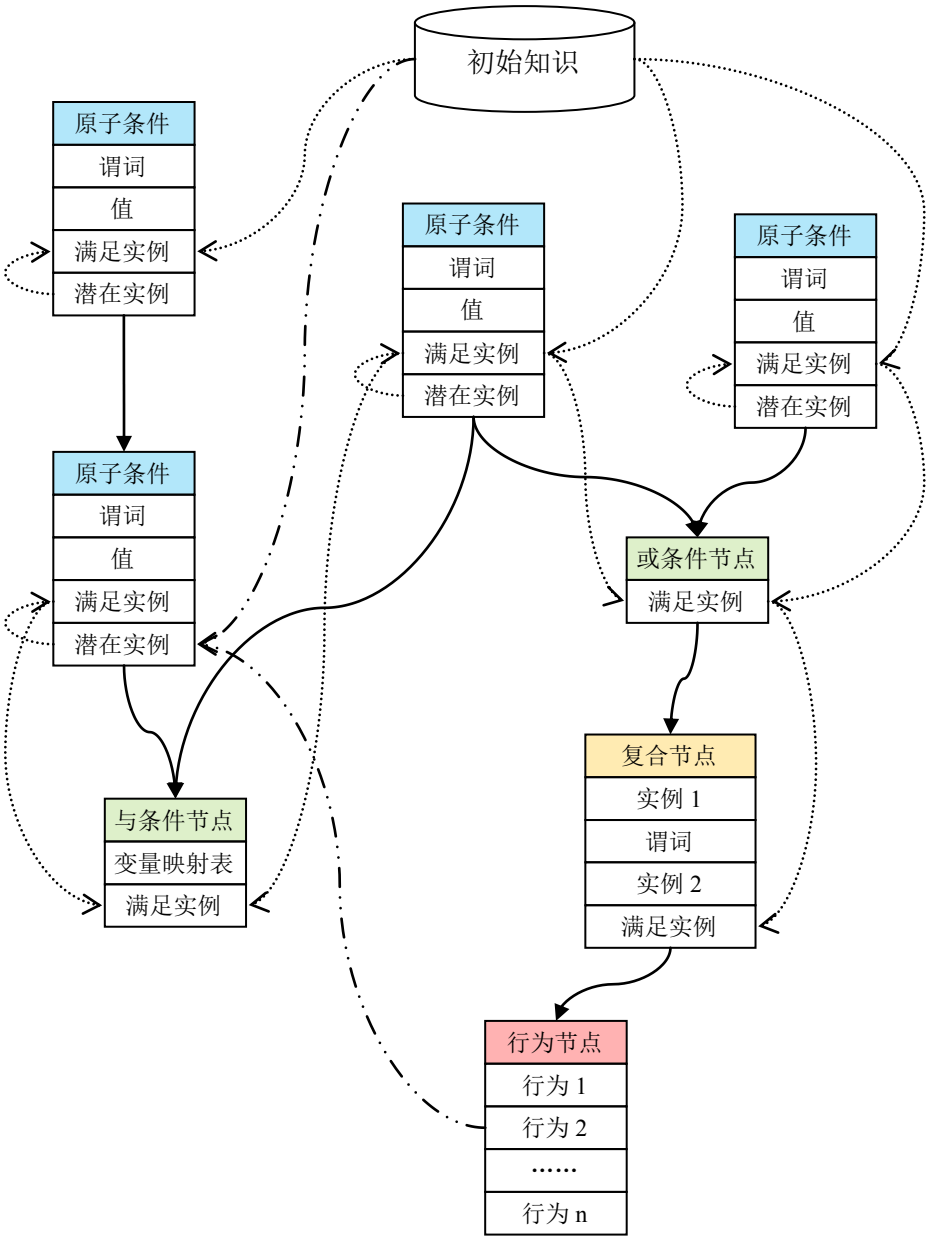

图 4-6 实例在各种节点间流动关系

Fig.4-6 Flow of Instances among Various Nodes

图 4-6 展示了实例在各种节点间流动的关系图。其中实线箭头图元代表从条件判断开始到标注行为实施的方向，这也是各个节点在网络图中的实际关联方向；虚线表示满

足条件的实例流动的可能性；点划线表示潜在满足条件的实例流动的可能性。

一开始，我们只拥有初始知识所包含的实例。对于这些实例：1）如果满足某个叶子原子条件节点，则该实例被送到相应原子条件节点的满足实例集合中；2）如果满足某个中间原子条件节点，则该实例被送到相应原子节点的潜在实例集合中。接下来，满足条件实例尝试向着实线箭头的方向进行流动，直到到达一个行为节点并触发该节点进行标注内容的产生。新产生的标注内容可能会使得原本不满足条件的节点成为满足条件的节点，也即从潜在满足条件集合中流动到满足条件集合中，并尝试继续往下流动。这个过程循环进行，直到没有更多的实例可以流动为止。

算法 4.1 将实现了上述思想。它根据一个给定的条件网络开始进行规则的匹配。该算法的参数 global-env 表示全局已有的本体知识，包括从 X3D 文档中提取出来的可视化信息以及标注人员已经完成的部分标注。该算法的初始部分 Fill-In-Initial-Knowledge 将初始知识分发到各个条件节点上，该算法将在 4.3.4 小节中讨论。

过程 4.1 和 4.2 分别为算法 4.1 调用的子过程。在该伪代码中：

move(x, s, t) 将节点 x 从集合 s 中转移到集合 t 中。

move-all(x, s, t) 将所有 x 集合中的节点从 s 集合中转移到 t 集合中。

valid(i, v) 检测实例 i 是否满足 v 节点所代表的条件。

is-atomic-condition(c) 等函数分别检测一个条件的各种类型

store(is, anno) 将 anno 所表示的标注中相关的实例存储到 is 集合中

find-condition(anno) 找出和 anno 相关的条件节点

4.3.3 规则条件网络的构造

对于一系列给定的规则，我们根据以下原则将它转换为一个条件网络：

0. 预处理：将所有规则中的 not 运算符利用逻辑运算分配到原子条件中，因此在网络图中我们不需要额外的“取反节点”，所有的取反操作都将在原子条件中表示。

1. 对于每一个表示类型的条件，例如(instance-of x XmlNode) ，我们为其创建一个原子条件节点，并且该节点在最后的网络图中作为叶子节点。

2. 对于每一个其余类型（非类型限定）的条件，我们为其创建一个相应的原子条件节点。这些节点在最后的网络图中通常为中间节点。

Algorithm 4.1: Match-And-Apply
Input: global-env
Input: condition-graph

```
1. let fireable-conditions ← Ø
2. Fill-In-Initial-Knowledge(global-env, partial-annotations, condition-graph);
3. while (fireable-conditions ≠ Ø) do
4.     let cond ← fireable-conditions.first();
5.     if (is-atomic-condition(cond)) then
6.         try-fire-valid-instance(cond, fireable-conditions);
7.         foreach (i, v) in {cond.Potential-Instances × cond.predecessors} do
8.             if (valid(i, v)) then
9.                 move(i, v, cond.Valid-Instances);
10.                fireable-conditions ← fireable-conditions ∪ cond;
11.            end if
12.        end foreach
13.    else if (is-or-condition(cond)) then
14.        try-fire-compound-condition(cond, fireable-conditions, ∪);
15.    else if (is-and-condition(cond)) then
16.        try-fire-compound-condition(cond, fireable-conditions, ∩);
17.    else then   // it is an action
18.        foreach action in cond.Actions do
19.            let new-annotation ← apply(action);
20.            let c ← find-condition(new-annotation);
21.            store(c.Potential-Instances, new-annotation);
22.            fireable-conditions ← fireable-conditions ∪ cond;
23.        end foreach
24.        foreach vertex in action.successor do
25.            fireable-conditions ← fireable-conditions ∪ vertex;
26.        end foreach
27.    end if
28. end while
```

Procedure 4.1: try-fire-valid-instance
Input: cond
Input: fireable-conditions

1. **if** (cond.Valid-Instances = Ø) **then**
2. fireable-conditions ← fireable-conditions – {cond};
3. **end if**
4. **foreach** (inst, vertex) **in** {cond.Valid-Instances × cond.successors} **do**
5. **if** (valid(inst, vertex)) **then**
6. move(inst, cond.Valid-Instances, vertex);
7. fireable-conditions ← fireable-conditions ∪ vertex;
8. **end if**
9. **end foreach**
10. fireable-conditions ← fireable-conditions – {cond};

Procedure 4.2: try-fire-compound-condition
Input: cond
Input: fireable-conditions
Input: op

1. try-fire-valid-instance(cond, fireable-conditions);
2. **let** instances ← op(cond.predecessors.map(p → p.Valid-Instances));
3. move-all(instances, cond.predecessors, cond.Valid-Instances);
4. **if** (instances ≠ Ø) **then**
5. fireable-conditions ← fireable-conditions ∪ cond;
6. **end if**

3. 对于相同变量的 and 组合，将 and 的元素创建出来的条件进行串联连接。对于规则 ((and (predicate1 x value1) (predicate2 x value2)) action) 将会构造如图 4-7 所示的网络：

图 4-7 同变量的与条件网络图

Fig.4-7 Condition Net for And Condition of the Same Variable

4. 在执行原则 1,2 的过程中，合并相同的节点。即对于相同的条件，只使用同一个

节点，不再创建新的节点。若存在以下两条规则：

((and (predicate-1 x 1) (predicate-2 x 2)) action-1)

((and (predicate-1 x 1) (predicate-3 x 3)) action-2)

将会被转换成如图 4-8 所示的网络图：

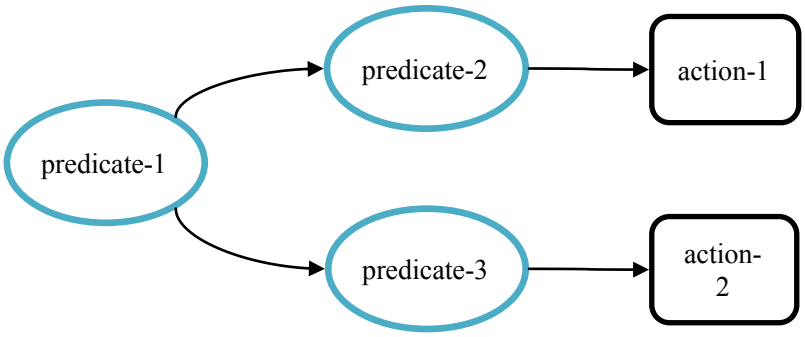

图 4-8 网络图中相同条件共用节点

Fig.4-8 Sharing of Node among Equivalent Conditions

5. 对于相同变量的 or 组合，分别为 or 的子元素创建不同的条件节点，并将它们并联连接。对于规则((or (predicate-1 x 1) (predicate-2 x 2)) action) 将会产生如图 4-9 所示的网络：

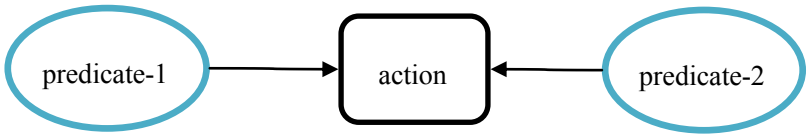

图 4-9 同变量的或条件网络图

Fig.4-9 Condition Net for Or Condition of the Same Variable

6. 对于不同变量的 and/or 组合，分别为 and/or 的子元素创建不同的条件节点路径，并将它们并联连接到相应的复合节点上。对于规则

((and (predicate-1 x 1) (predicate-2 y 2)) action)

将会产生如图 4-10 所示的网络：

7. 对于每个规则行为，为其创建行为节点，同样的，合并相同的行为节点。

8. 为每一组相同的规则条件和规则行为，假设已分别为它们创建条件节点 x 和行

为节点 y，则继续创建从 y 到 x 的回边。

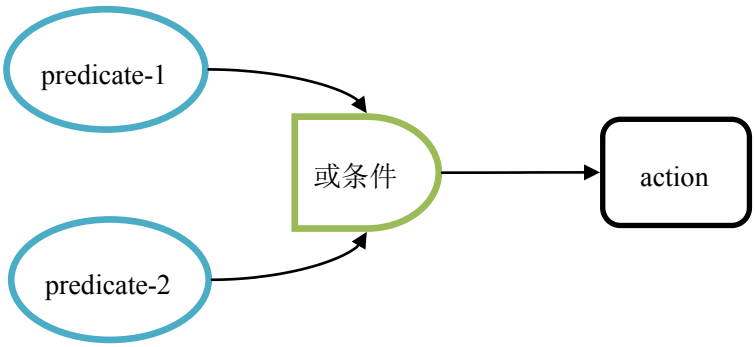

图 4-10 不同变量的复合条件网络图

Fig.4-10 Condition Net for Compound Condition of Different Variables

4.3.4 初始知识的分配

由第三章所提出的算法得到的可视化信息的数量可能是极其庞大的，远远超过标注过程所实际需要。事实上，很多可视化信息在标注过程中不会被用到。这体现在：1）许多的物体节点并不真正表示一个在人眼看来有意义的物体；2）许多物体的属性在规则的条件描述中完全没有被用到；3）许多物体的相互位置关系在规则的条件描述中没有被用到。然而，可视化信息提取模块由于对标注过程的信息未知，会将所有可能得到的知识获取出来。因此，我们在标注过程开始前所拥有的初始信息比实际需要的要大，如果将它们全部输入进条件网络，会造成许多不必要的计算，从而降低整个标注过程的效率和性能。

在本文的研究中，我们采取分析条件节点并从大量的初始知识中拉取（相对于由可视化信息抽取模块像标注引擎推送的做法）有用知识的策略。具体来讲，每一个原子条件节点都用本体属性的方式表示了一个需要满足的事实。对于每一个这样的节点，我们使用本体推理机进行推理，在大量知识库中找到符合条件的实例，并将它们送到那个节点的满足条件实例集合中。这样做的好处是可以利用本体推理机本身的算法优势，避免手动地去遍历每个知识然后在条件网络中寻找匹配的条件节点。

4.4 规则一致性验证

规则的一致性所表达的含义是，一系列规则不会导致互相矛盾或冲突的标注内容产生。但是实际上，该定义是较为模糊的。例如，一个标注规则指示将一个物体既标注为桌子又标注为房间，则从常识的角度来讲，我们可以确定这是相互冲突的。然而对于标

注引擎来讲，它必须知道桌子和房间是完全不同的概念才能做出这样一种判断。

本文关注的规则一致性主要集中在以下两个方面：

1. 对于单独的一个规则，它所包含的若干标注行为之间不会产生冲突情况。

2. 对于多个规则并存的情况下，这些规则不会循环导致彼此被触发。

Algorithm 4.2 Condition-Combinations
Input: rule
Input: global-knowledge

```
1. let condition ← rule.condition;
2. let actions ← rule.actions;
3. let comb-set ← Condition-Combinations(condition);
4. foreach cond-set in comb-set do
5.     apply-all(cond-set, global-knowledge);
6.     let result ← Check-For-Ontology-Inconsistency(global-knowledge);
7.     if (result = true) then
8.         return true;
9.     end if
10.    unapply-all(cond-set, global-knowledge);
11. end foreach
12. return false;
```

Procedure 4.3 Condition-Combinations
Input: condition

```
1. if (is-atomic-condition(condition)) then
2.     return condition;
3. end if
4. if (is-and- condition (condition)) then
5.     return ∪c ∈condition.sub−conditions[Condition − Combinations(c)];
6. end if
7. if (is-or-condition(cond)) then
8.     return ∏c ∈condition.sub−conditions[Condition − Combinations(c)];
9. end if
```

对于上述第一种情况，本节引言中的例子就是一个很典型的冲突情况。为了能够检测出该状况，我们的本体中必须明确申明桌子和房间是相互互斥的概念。这个做法带给

我们两个启示。首先，如果我们的本体是严格定义的，那么通过本体推理，我们就可以查询出不一致的知识集合。然而另一方面，在实际应用中，我们很难期望本体的定义是如此严格以至于将每两个概念都明确地用本体语义指明它们之间有无相关性。因此这种检测手段在一定程度上取决于本体本身的严格程度。现在假设本体是部分严格的，则我们可以通过算法 4.2 来验证若干规则行为的一致性。该算法过程接受一个规则作为输入，返回该规则是否拥有冲突的规则行为。

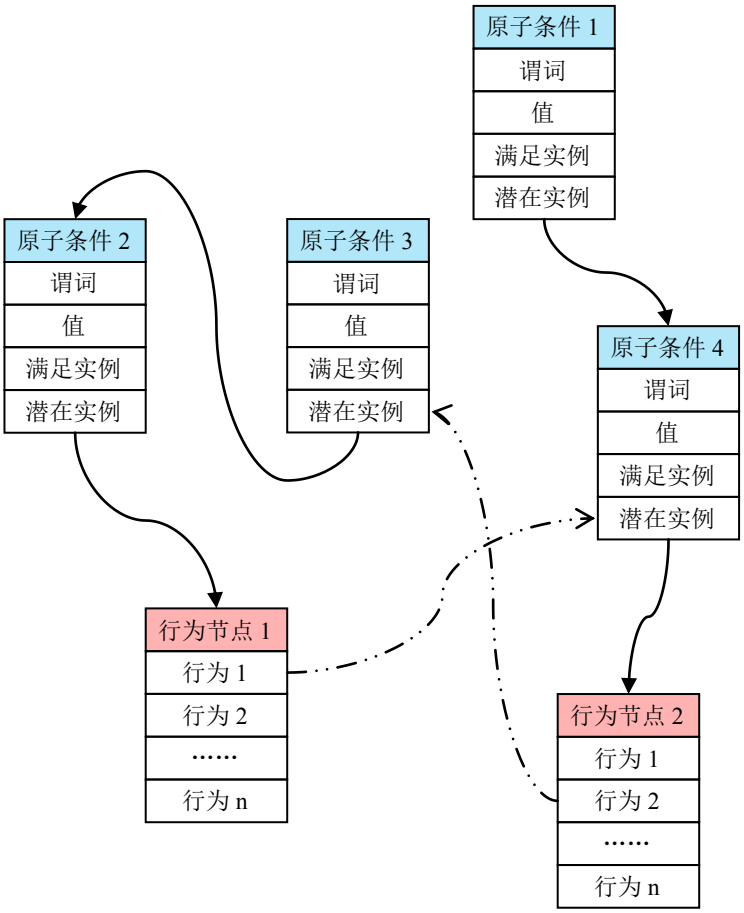

图 4-11 循环触发规则示例

Fig.4-11 Example of Circular Firing of Rules

对于上述第二种情况，我们需要利用到之前介绍的条件网络图。我们所要确定的是在该网络图中是否存在环。如果存在环，则表示不同的条件之间有可能产生循环触发，从而使得整个标注过程无法正常中止。如图 4-11 所示就是一个具有环的条件网络图，该网络可能导致这样的触发循环：条件 4 → 行为 2 → 条件 3 → 条件 2 → 行为 1 → 条件 4. 以之前提到的会议桌和会议室的例子来讲，用户将不能同时定义它们的双向关

系，而只能定义其中单向的标注规则。在实际应用中，这样的双向规则很可能是需要的，因此在后续的研究中，我们将把注意力从避免循环上移开，转向循环的探测和动态排除，使得从图论角度看，静态的条件网络可能存在环，但在实际标注中，我们可以检测到循环触发并在适当的时候中止它。

4.5 本章小结

本章详细介绍了如何利用规则的定义和实施来抽象语义标注的过程。在描述如何构造合法的标注规则表达式的同时，本章还说明了自动标注引擎的工作流程以及它和本体推理的关系。随后，本章着重阐述了规则集合的网络图表示结构，以及如何用它来解决大量场景中快速完成规则匹配的算法。最后，本章讨论了如何从两个方面检测用户定义规则的不一致性。

5 系统实现与验证

本章将具体介绍面向 X3D 基于本体的自动语义标注系统的实现与验证。本章将首先给出原型系统的总体架构，随后对其中的部分模块的实现细节进行讨论。本章的后半部分将通过一个具体的应用实例来介绍如何使用我们的系统，并由应用的结果来验证本文所提出方法的实用性和有效性。

5.1 半自动语义标注框架总体架构

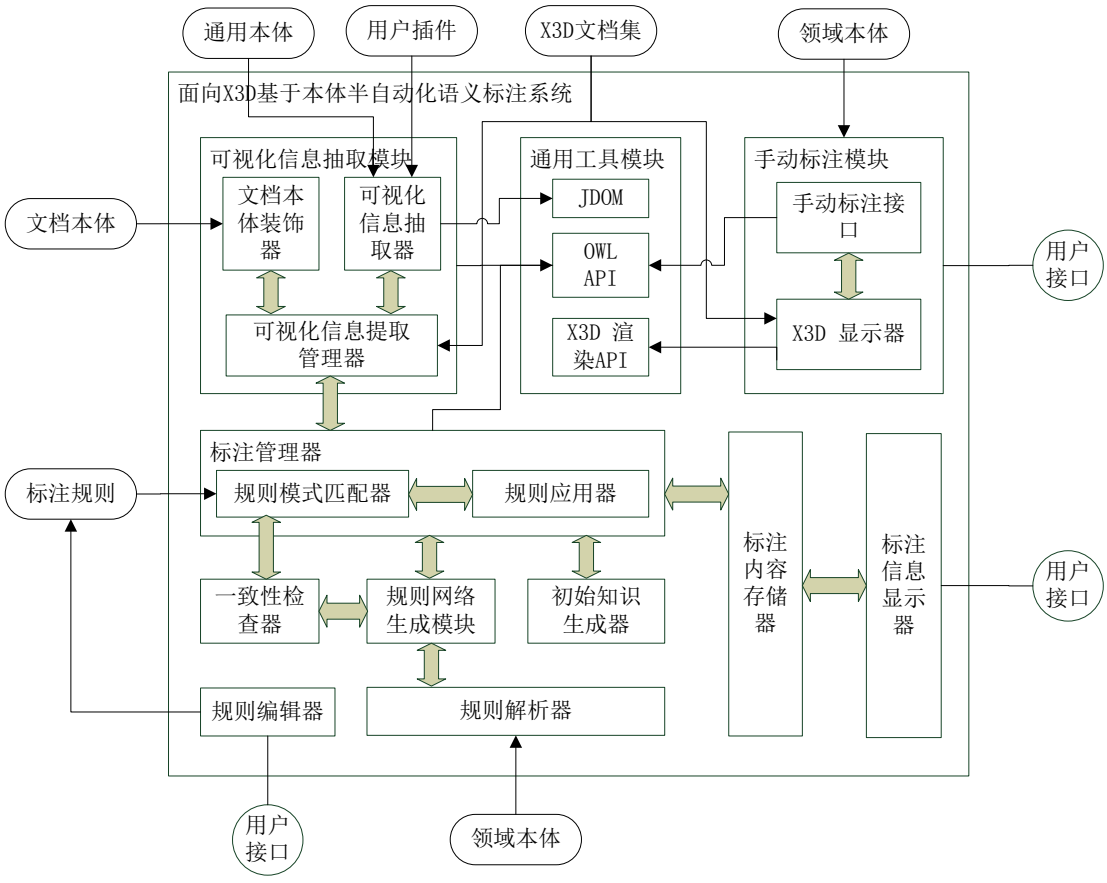

图 5-1 半自动化语义标注系统架构

Fig.5-1 Architecture of Semi-Automatic Semantic Annotation System

图 5-1 所示是本文提出的半自动化语义标注框架的系统实现总体架构。最外围的矩形框代表了整个系统的边界，在此边界之外有作为系统必要输入的 X3D 文档集合、各

种类型的本体文件、标注规则；由标注人员提供的部分标注；以及供各类用户查看的X3D 和标注内容显示接口。

该原型系统采用 Java 作为实现平台，并最终实现为一个应用系统。在图中，我们可以看到在通用工具模块中有着 Java 语言用于处理 xml 以及本体的类库。我们的系统依赖于这些第三方工具来方便地完成底层的功能。这些第三方工具包括用于支持 XML 操作的 JDOM[33]类库，用于支持本体操作和知识存储的 Jena[34]库和 Sparql[35]查询语言，以及用于 X3D 显示的 xj3d[36]工具[37]。用户可在该系统上建立标注工程，并为每个工程设定独立的通用本体，领域本体以及文档本体。在每个工程中，用户可制定一系列标注规则，随后导入若干 X3D 场景文件，以及部分标注，来开始自动标注的流程。

5.2 标注系统各模块实现

5.2.1 可视化信息抽取模块

在图 5-1 所示的架构左上方，我们绘制出了可视化信息抽取模块内部的主要结构，从中可以看到，它主要由可视化信息提取管理器，可视化信息抽取器和文档本体装饰器三部分组成。其中可视化信息提取管理器负责与外部接口和系统内其他模块的相互协作，以及其自身内部子模块的调度工作，来完成第三章介绍的 X3D 文档可视化信息提取工作。本小节将主要介绍其中的两个实现细节。

第一为文档本体装饰器组件。在第三章的介绍中，我们从 X3D 中提取的知识完全以通用本体作为载体呈现。然而为了使得用户也能够用 XML 节点的特征作为标注规则的一部分，我们引入了文档本体这一概念。这也就意味着，作为自动标注输入的初始知识的一部分，从 X3D 场景文档中提取出来的可视化信息也需要能包含用文档本体所描述的知识。这些知识当可视化信息提取操作完毕的时候已经完全存在了，只不过它们是存储为通用本体的形式。因此，在此基础上，我们建立了文档本体装饰器组件，它将可视化信息提取器提取出来的可视化信息作为输入，并将它们转换为以文档本体表示的知识。

具体来说，对于可视化信息中每一个本体实例 x，我们为其生成一个对应的文档本体实例 y，属于 XmlNode 概念类。这两个实例将被文档本体中的 hasNode 和 hasInstance 等属性关联起来。对于每一个这样建立的 XmlNode 实例，

1）我们检测它的节点属性，并用 hasAttribute 属性关联起来；

2）检测它们之间的相互关系，并用文档本体中 hasSibling 等属性作出关联。

第二为可视化信息抽取器采用用户插件的方式实现。在第三章可视化信息提取的思想介绍中，我们提到即便对于一个最简单的概念（如色彩），它的实际值也会包含在 X3D 节点的不同地方，因此我们需要策略去获取它。事实上，对于每一个在通用本体中存在的概念，我们都需要为它们定制各自的获取策略。这个策略我们希望可以动态改变，或者至少静态可配置，因为我们采取了用户以插件方式自定义这些策略的实现方式。图 5-2 展示了可视化信息抽取模块的主要类的架构图。

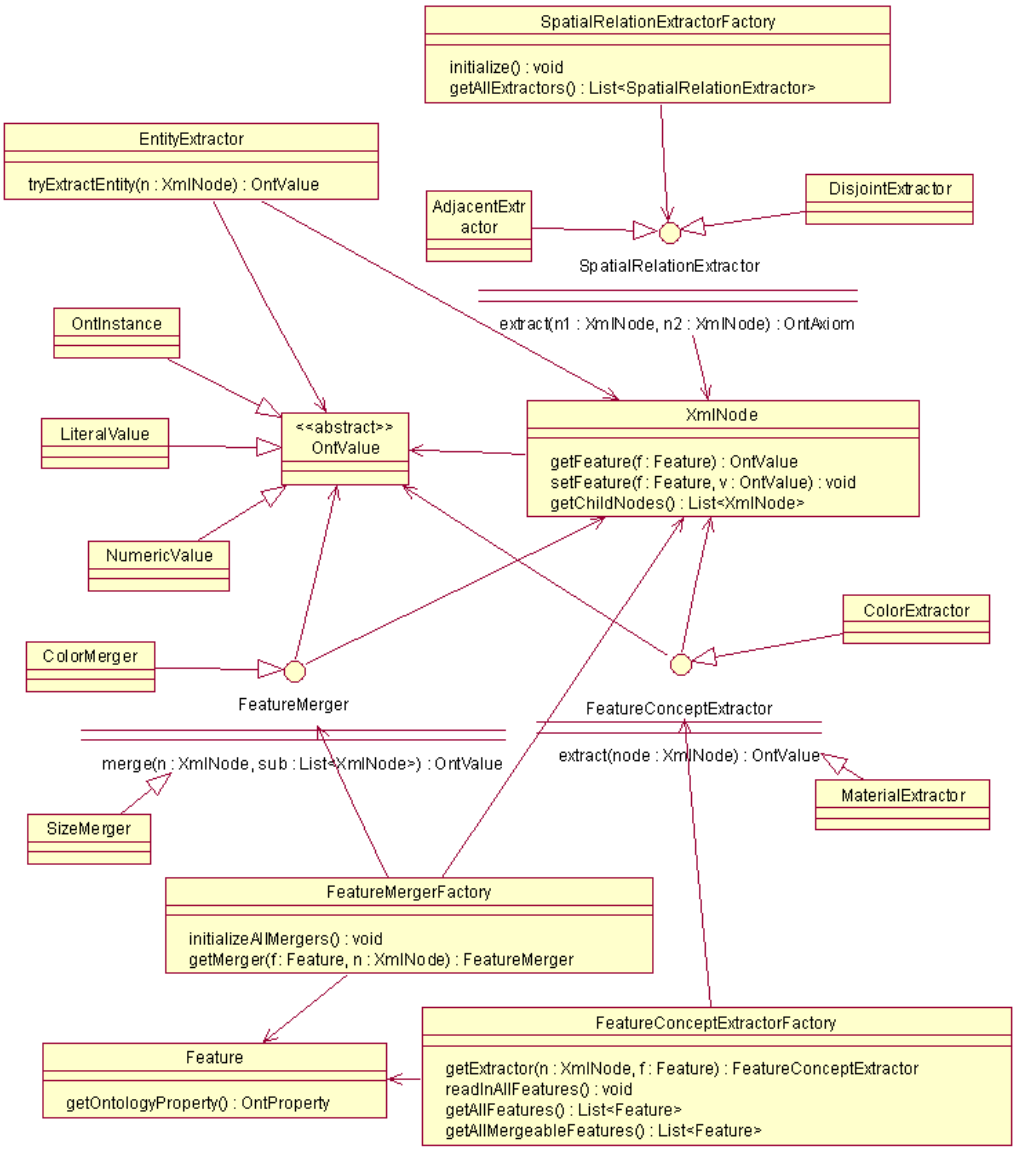

图 5-2 可视化信息抽取模块主要类图

Fig.5-2 Class Diagram for Visual Information Extraction Module

在该架构中，我们为每一种可配置的抽取策略定义了一个工厂。例如对于特征概念的抽取，我们定义了 FeatureConceptExtractionFactory 工厂类，它在系统初始化的时候将读入配置文件，来得到哪一种特征概念对应于哪一个抽取类(FeatureConceptExtractor)的信息，并将后者实例化放入一个映射表中，以便后续查询。对于 FeatureConceptExtractor 这样的接口，每一个该接口的实现类都表示对于一种特定的本体概念的抽取策略。用户可以通过自行实现接口，并在配置文件中标出新实现类的名字这样的方式来覆盖上述接口的默认实现，以达到自定义的目的。

图 5-3 展示了可视化信息提取过程中 VisualInfoExtractorManager 类如何调用各个工具类的过程。该图中为了简便将 FeatureConceptExtractor 简写成 FCE。该过程对应于 3.4 节中提出的算法，并用到了图 5-2 中展示的类和它们所拥有的方法。

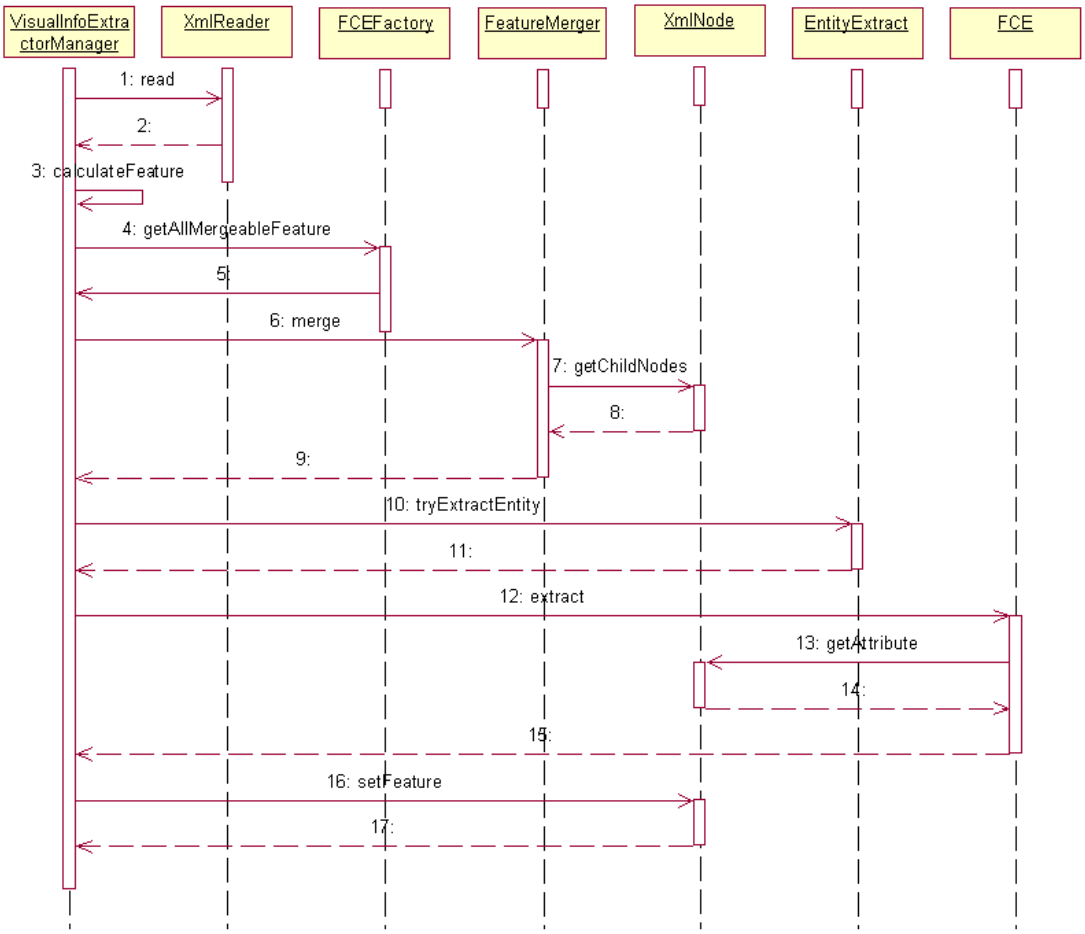

图 5-3 可视化信息计算及合并顺序图

Fig.5-3 Sequence Diagram for Computing and Merging of Visual Information

5.2.2 标注规则模块

图 5-4 展示了规则相关模块的部分类图。在该类图中，我们展示了 1）规则类(Rule)如何细分为条件类(RuleCondition)和行为类(RuleAction)来表示；2）通过继承的方式来定义不同类型的条件(AtomicCondition 和 CompoundCondition)；3）从文本规则到规则对象的解析过程中的类交互(RuleReader 和 RuleParser)。在该架构中，RuleObject 类存在自引用，从而实现了用于表达规则的列表对象的递归形式。

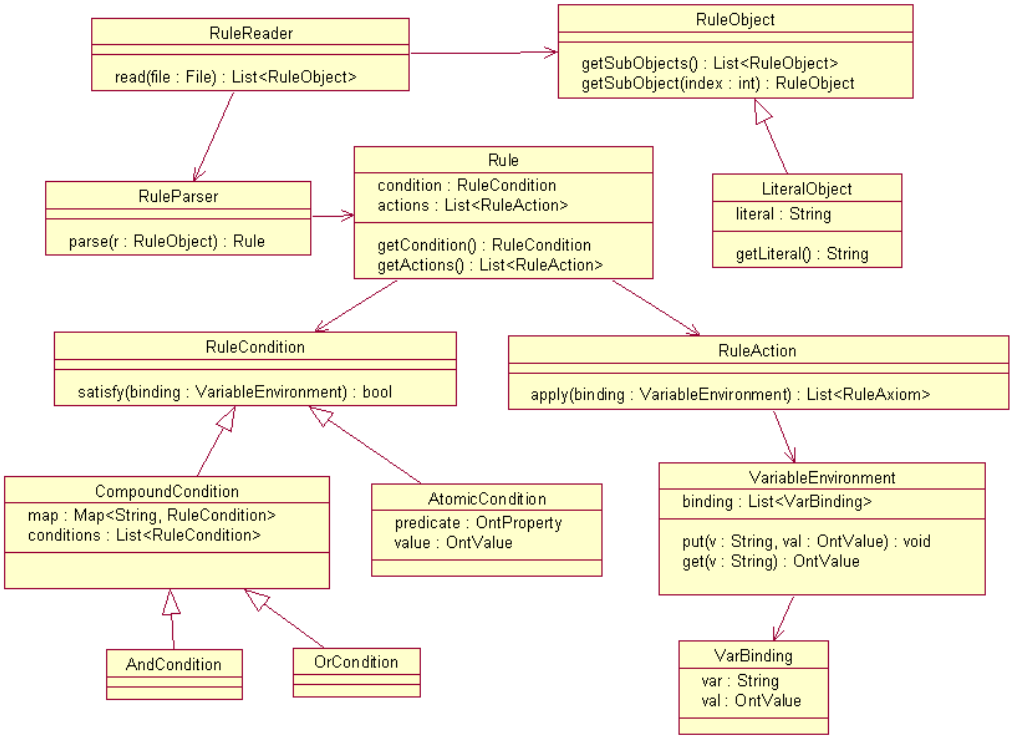

图 5-4 规则读取模块类图

Fig.5-4 Class Diagram for Rule Interpretation

图 5-5 展示了规则相关模块中匹配和应用部分的类图。在该架构中，SemiAutoAnnotator 是整个标注过程的控制类，它调用其他工具类来完成标注工作。其中 ConsistencyChecker 用于检查规则的一致性，它包含方法接受一组规则，返回其是否具有一致性。ConditionNetBuilder 负责将用户输入的规则转换成条件网络。从 VisualInfoExtractorManager 类得来的可视化信息将被用于构建该网络中使用的初始知识。AnnotationPersister 负责将由标注引擎生成的标注持久化到相应的数据库中。NetNode 是一个抽象类，表示所有在条件网络中存在的节点，它的子类利用组合模式定义了不同种类的条件网络节点。

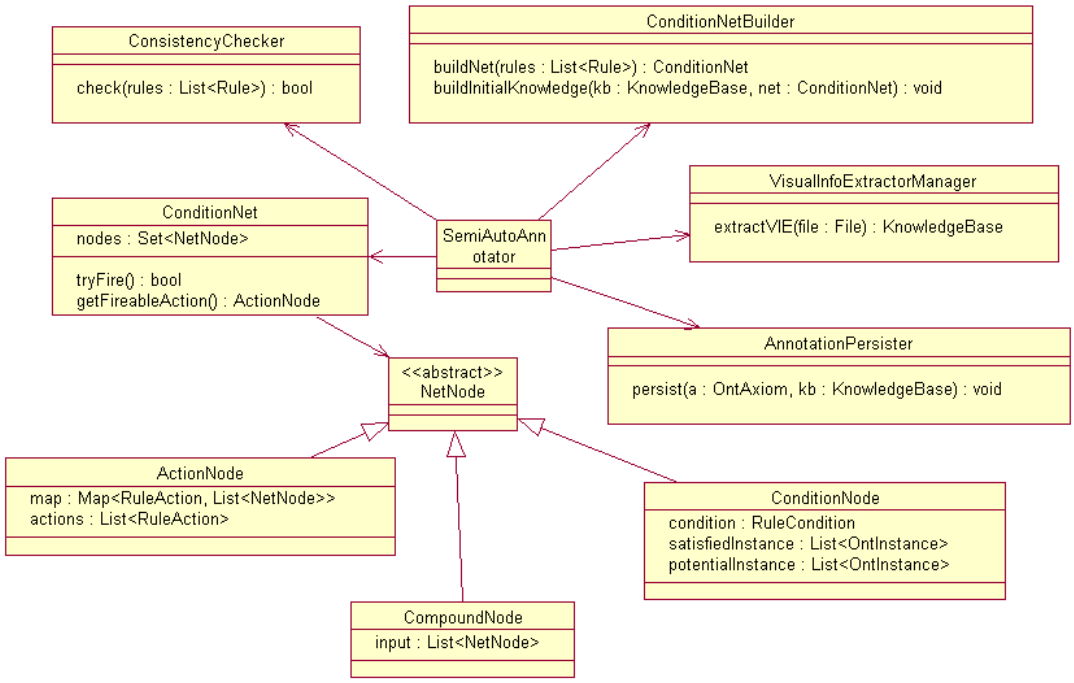

图 5-5 规则匹配与应用部分类图

Fig.5-5 Class Diagram for Rule Matching and Application

5.2.3 外部输入的创建

本系统共用到四种外部输入，分别是作为标注对象的 X3D 文档，作为标注内容载体的各类本体，作为自动标注依据的标注规则，以及作为自动标注初始知识一部分的部分标注。对于作为标注对象的 X3D 文档，我们假设建模人员已经利用已有的实体建模工具完成建模。对于作为标注内容载体的各类本体，我们假设领域专家已经利用文本编辑工具或可视化本体编辑工具（如 Protégé[38]）完成创建。

对于作为自动标注依据的标注规则，我们的系统提供了标注规则编辑模块，除了提供最基本的文本编辑以外，还提供了可用的变量，可用的属性等快捷提示功能，方便规则的快速创建。对于作为初始知识一部分的部分标注，我们使用图 5-1 右上方的手动标注模块。该模块的实现将为用户可视化的展现场景，并附以该场景的 XML 文档以及标注过程中用到的本体，用户可以在文本中选择一个 X3D 节点，并将它关联到某个本体实例，或者为某个本体实例关联它的属性和属性值，来完成部分标注。

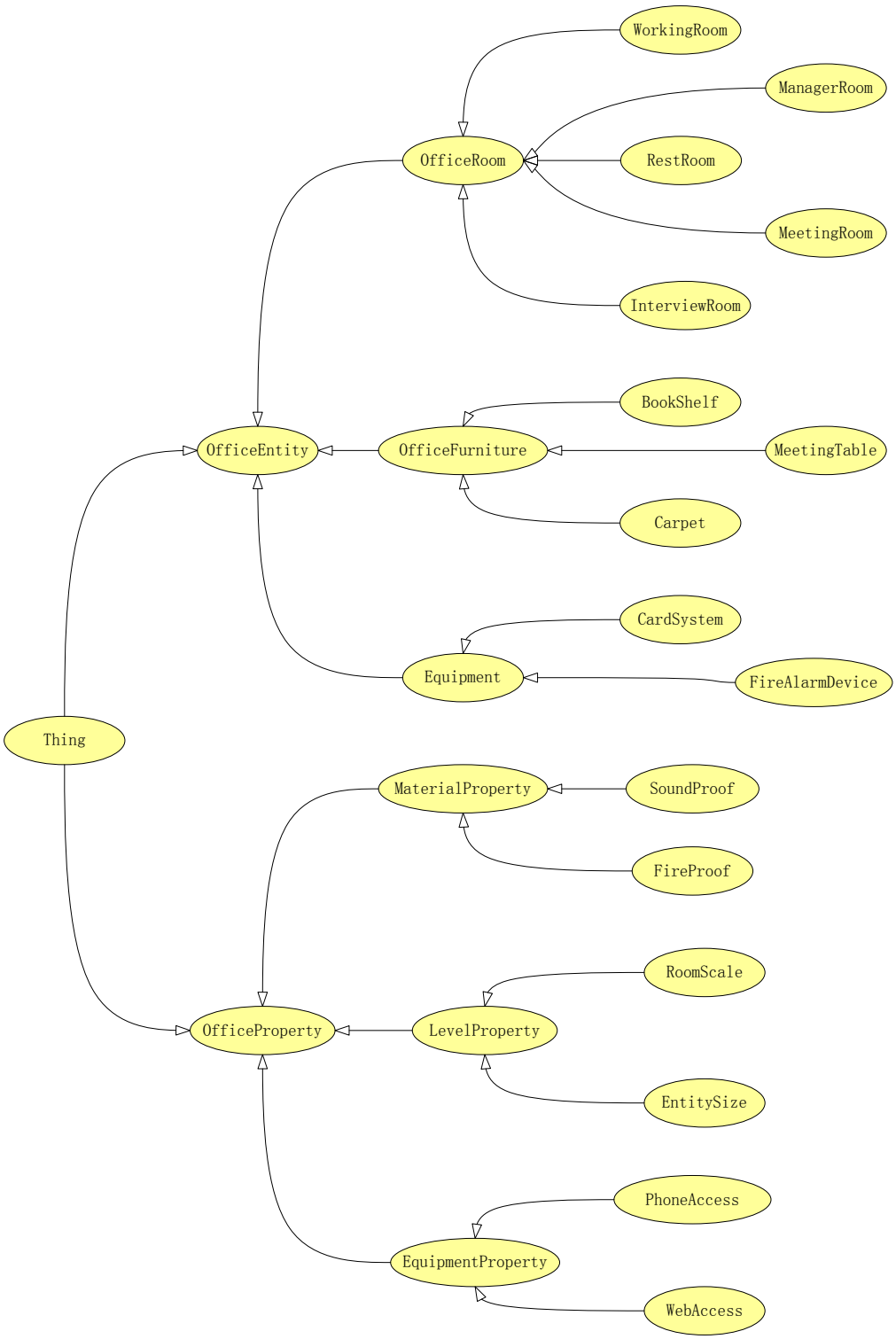

图 5-6 领域本体样例

Fig.5-6 Example of Domain Ontology

5.3 标注系统的应用与验证

本小节将使用办公楼作为应用场景，以一个用例来演示如何应用本文提出的标注系统对 X3D 场景进行半自动语义标注。我们将根据标注的流程一一完成标注所需的输入，最后利用半自动标注引擎进行标注内容的生成。

5.3.1 本体的创建

本文所提出的标注系统需要的本体有三类，即通用本体，文档本体以及领域本体。我们假设在前两者已经分别用第三第四章中所介绍的方式构建完毕。而对于办公楼的场景建模，我们则需要额外建立领域本体。一个样例本体如图 5-6 所示，其中包括了楼内室内信息以及基本的办公用品。

5.3.2 规则的创建

针对一个典型的办公楼场景，我们识别出一个房间内的设备构造和该房间的性质存在着一定的联系。作为样例，我们编写了如下的标注规则：

```
((and    (lists-of chairs c) (instance-of c MeetingChair) (instance-of t Object)
         (surround chairs t))
 ((instance-of t MeetingTable))

((and    (lists-of chairs c) (instance-of c MeetingChair) (count chairs num)
         (instance-of t MeetingTable) (instance-of r Object) (greater-than num 10)
         (direct-contains r t) (direct-contains r chairs)(at-the-center t r))
 ((instance-of r MeetingRoom)
  (hasRoomScale r (ont-instance `ontology-uri#large-room`))

((and    (x contained-in y) (y instance-of MeetingRoom))
 (x in-room y)
```

在该文本中我们定义了三条规则。在第一条规则中，我们将被多把座椅围绕的物体定义为会议桌，其中用到 list-of 操作符来定义一个集合的对象。在第二条规则中，我们将拥有在其内部中心拥有会议桌并且具有一定数量座椅的空间定义为会议室，并将后者的规则设置为大型。在第三条规则中，我们将会议室的包含关系转换为领域本体中的

in-room 属性。在该规则中，我们用到了 greater-than 基本操作符来约束数量，用到了 direct-contains 和 at-the-center 两个通用本体中的属性来约束空间关系。最后我们用 ont-instance 操作符来获取由指定字符串所引用的本体实例。

在这三个简单的规则定义中，我们用到了标注的反馈机制。即，由标注引擎自动生成的标注内容可以反过来触发更多的规则，产生更多的标注。在这个例子中，第一个规则产生的 MeetingRoom 本体实例可能导致第二个规则的触发。在实际的条件网络图中，代表规则 1 的行为节点将有一条回边指向规则 2 的若干原子条件节点。

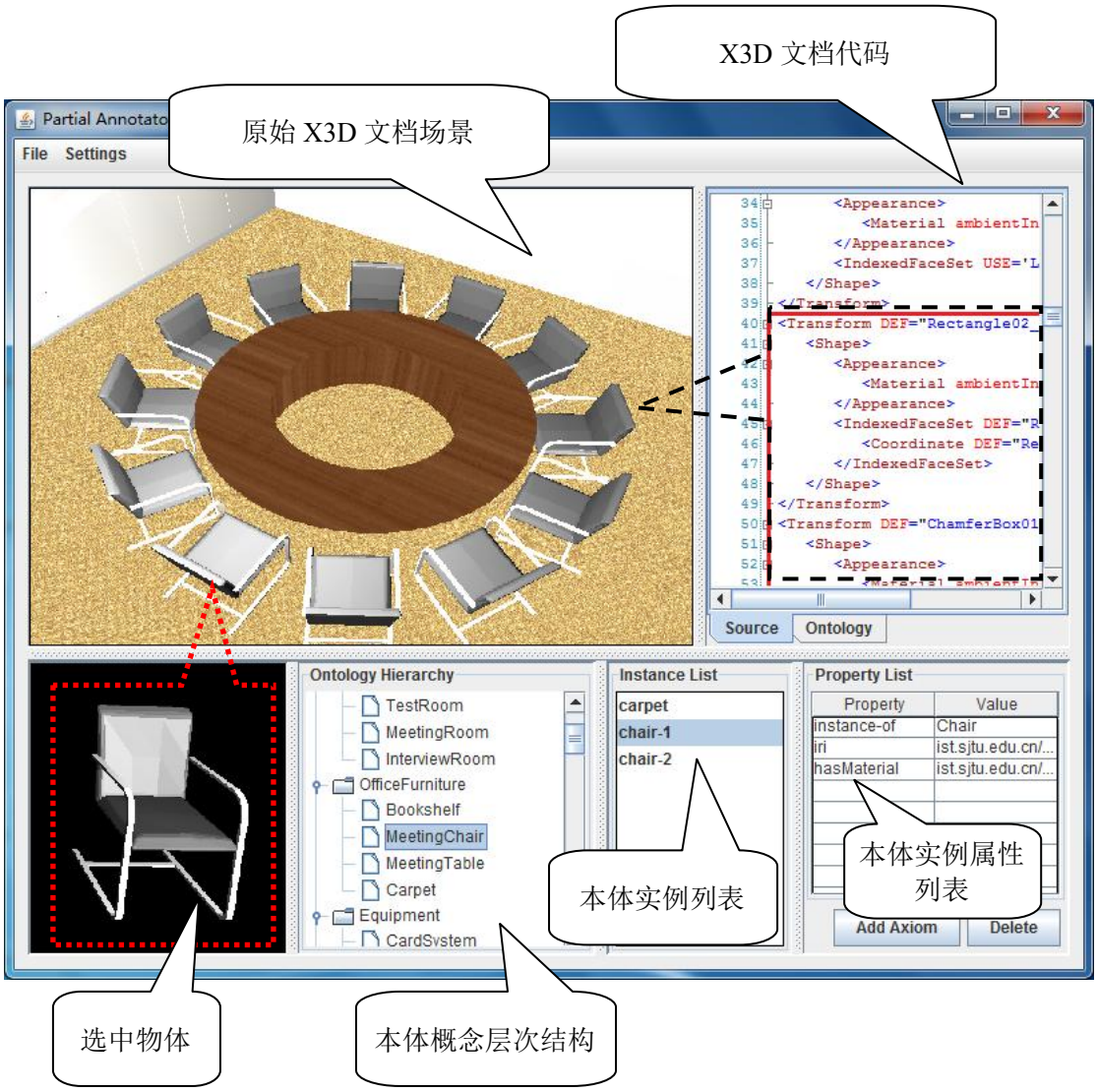

图 5-7 部分标注的创建

Fig.5-7 Construction of Partial Annotation

5.3.3 部分标注的创建

作为上述规则的验证，我们需要选取一个会议室的场景文档，并对其中的一把座椅进行标注。该过程的产物将作为部分标注输入到自动标注引擎中。图 5-7 是我们系统的运行截图，该截图展示了用户用来进行部分标注的操作界面。图示的左上方显示了所导入的 X3D 文件描绘的场景，图示的右上方则显示了其对应的 XML 源代码。用户通过选择源代码中的节点来浏览物体，选中的物体会在界面左下方的区域显示出来。当用户选定一个物体后，可以通过界面下方中央的本体树中选择本体概念，关联到之前所选的物体上。在这个绑定完成后，用户还可以通过界面右下角的“添加知识”按钮为新创建的本体实例添加属性，从而完成部分标注的工作。作为本节的示例，我们将其中的一把座椅和本体概念 MeetingChair 关联起来。

5.3.4 自动生成标注内容

当完成上述工作后，我们就可以开始启动标注引擎自动标注的过程。针对上述例子，其标注的结果如图 5-10 所示。该图所示的软件界面上方为待标注的 X3D 文档渲染画面，下方为系统生成的标注内容。这些标注信息中包括场景内其余椅子对象，会议桌对象和会议室对象的概念识别，以及它们间的相互关系。它们详细信息如图 5-8 和 5-9 所示。

Subject	Property	Value
instance_118:MeetingChair	in-room	instance_116:MeetingRoom
instance_118:MeetingChair	has-xml-node	meetingroom.x3d/x3d/Transform[4]
instance_119:MeetingChair	in-room	instance_116:MeetingRoom
instance_119:MeetingChair	has-xml-node	meetingroom.x3d/x3d/Transform[5]
instance_120:MeetingChair	in-room	instance_116:MeetingRoom
instance_120:MeetingChair	has-xml-node	meetingroom.x3d/x3d/Transform[6]

图 5-8 生成的标注内容前半部分

Fig.5-8 First Half of the Generated Annotation Content

instance_128:MeetingChair	in-room	instance_116:MeetingRoom
instance_128:MeetingChair	has-xml-node	meetingroom.x3d/x3d/Transform[14]
instance_116:MeetingRoom	has-xml-node	meetingroom.x3d/x3d/Transform[1]
instance_117:MeetingTable	has-xml-node	meetingroom.x3d/x3d/Transform[20]
instance_117:MeetingTable	in-room	instance_116:MeetingRoom
instance_116:MeetingRoom	has-table	instance_117:MeetingTable
instance_116:MeetingRoom	has-chairs	12:Integer
instance_116:MeetingRoom	has-room-scale	instance:#large-room

图 5-9 生成的标注内容后半部分

Fig.5-9 Second Half of the Generated Annotation Content

图 5-8 展示了标注内容的上半部分。可见该部分主要包含了（1）椅子对象的类型

识别，以及这些对象在 X3D 文档内的节点和本体实例的对应关系；（2）椅子对象和会议室对象的语义关系（来源于上一节标注规则的最后一条）。图 5-9 展示了标注内容的下半部分。该部分主要包含了（1）会议桌和会议室对象的识别和它们关于本体实例的关联；（2）会议桌与会议室的相互关系；（3）根据上一节第二条标注规则生成的关于会议室规模的知识。

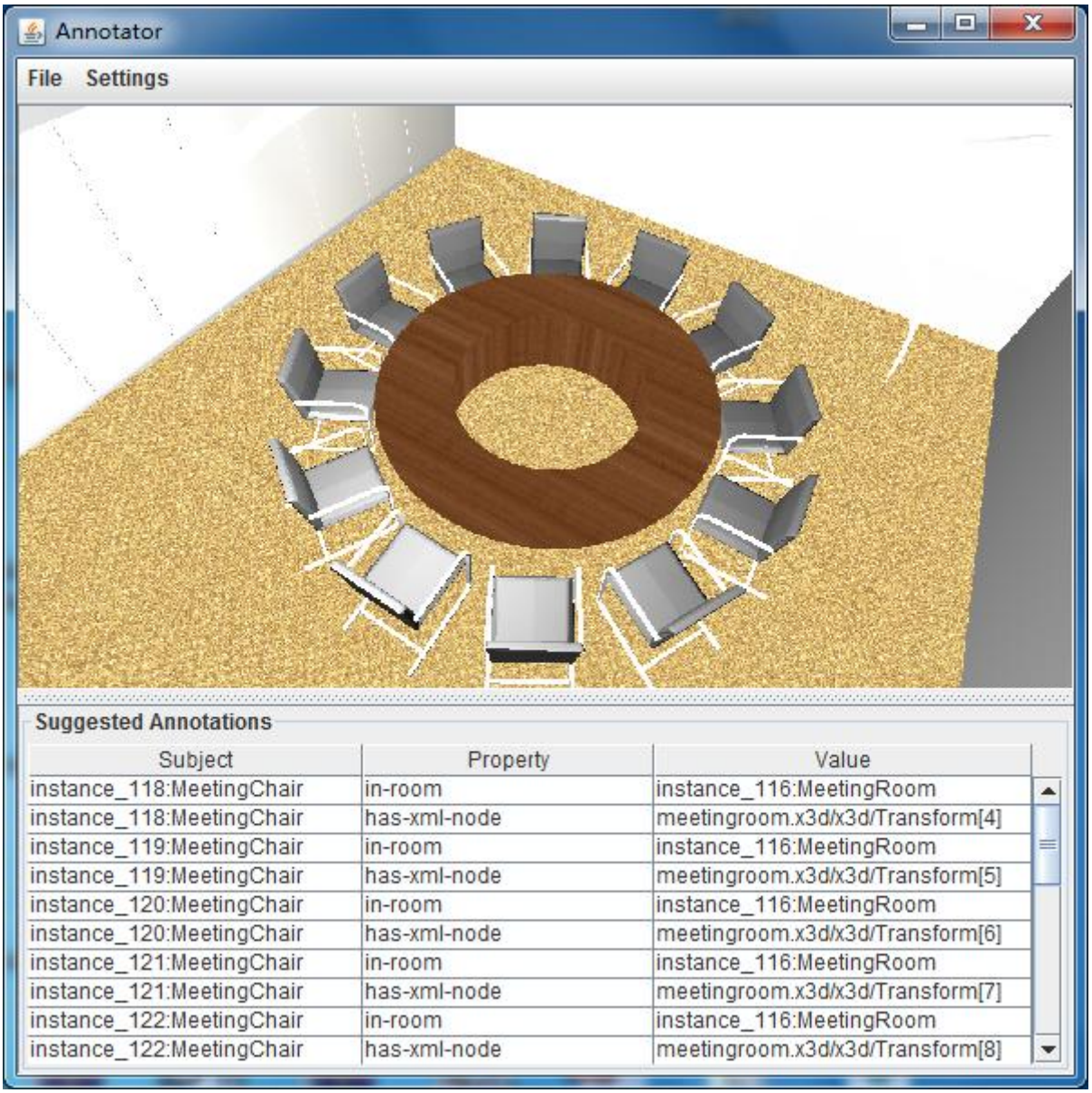

图 5-10 半自动标注结果

Fig.5-10 Result of Semi-Automatic Annotation

5.3.5 物体间相互关系的抽取

上一小节中的软件界面展示了系统将会生成哪些语义标注。这些标注的生成过程中依赖于系统前期对三维文档的处理，也即第三章介绍的可视化信息抽取。图 5-11 给出了上述应用例子中所抽取出的物体间相互关系图。该图由我们的自动标注原型系统自动生成。该功能根据所选中的物体，为我们提供与此物体相关的物体间关系。

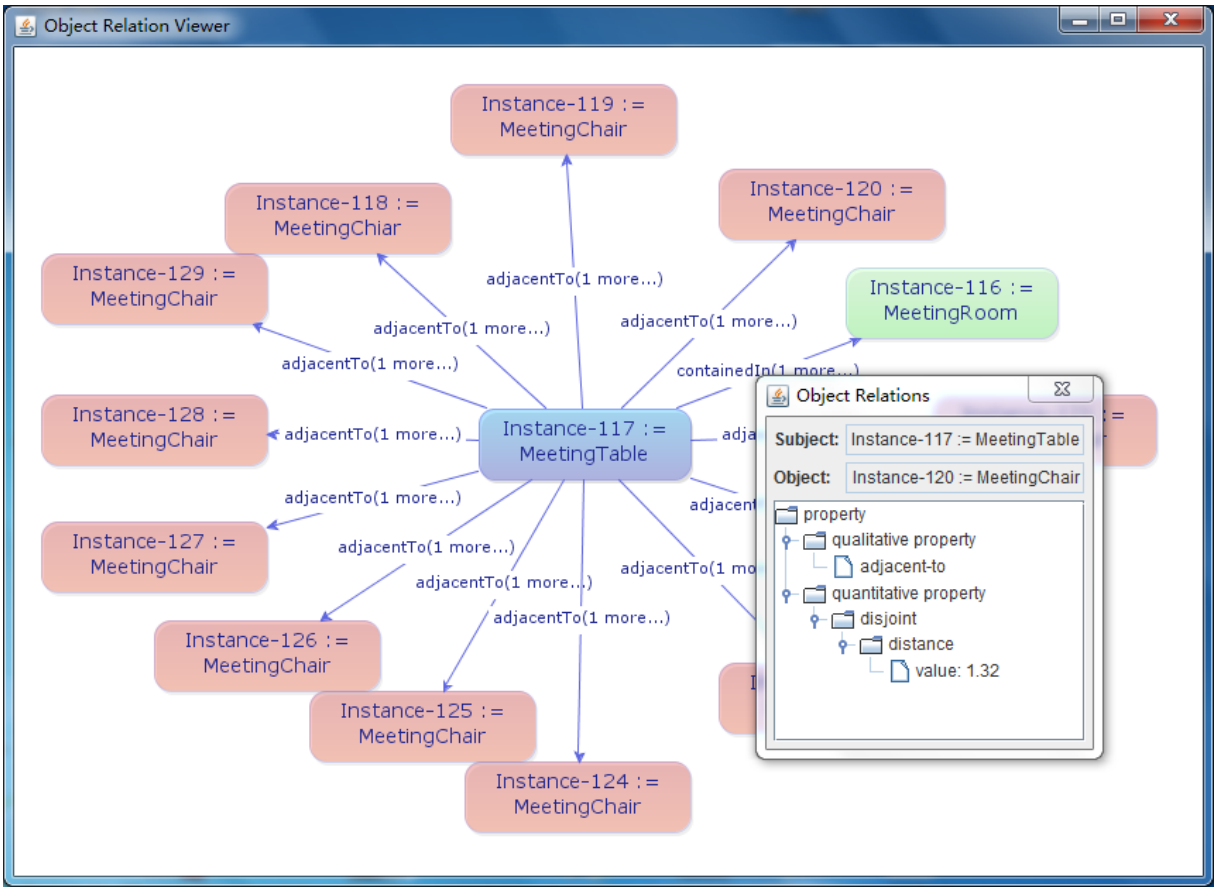

图 5-11 物体间关系图

Fig.5-11 Graph of Inter-Object Relationships

我们选取了会议桌作为该图的演示对象。因此该对象出现在图中央，其四周和其连接的图元则代表被抽取出来和它有相互空间关系的对象，这些对象包含了会议室内的椅子和会议室本身。在它们的连线上标出了它们的相互关系。任意两个物体间可能存在多个关系，详细查看这些关系将会打开图示中的对象关系对话框。图 5-12 展示了两个这样的对话框。它们分别表示会议桌对象和其中一把椅子对象，以及会议桌对象和会议室对象所存在的相互关系。

在左边的对话框中，我们看到的是会议桌和某一把座椅的关系。该关系中包含了第三章中提到的定性属性以及定量属性。其中桌椅间的相邻关系为定性属性，它们间的相

离关系则是更底层的带有距离数值的定量属性。右边的对话框所展示的则是会议桌与会议室的关系。该关系包含了包含和处于中央等两个定性属性，但没有定量属性。

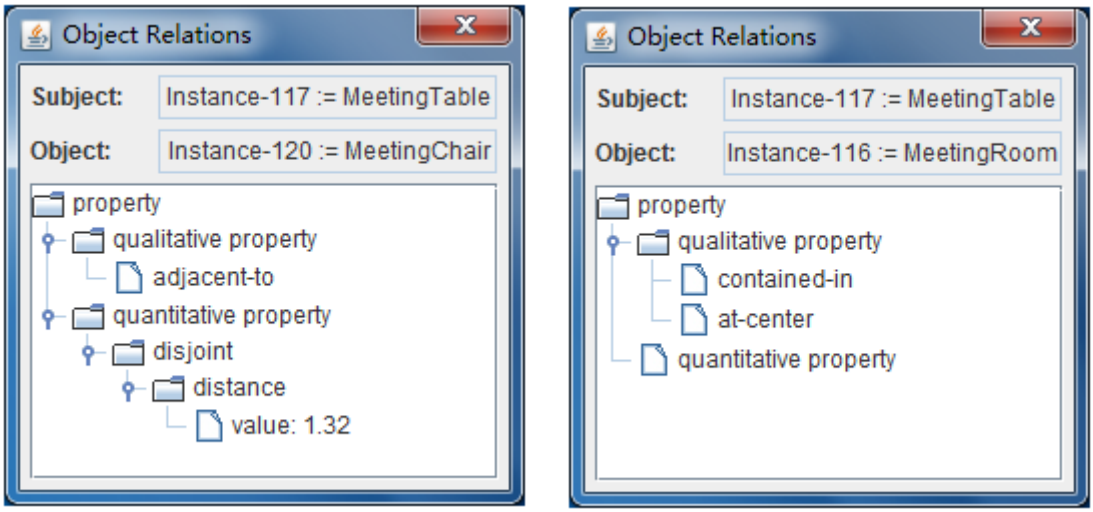

图 5-12 物体间关系详细信息

Fig.5-12 Detail Information of Inter-Object Relationships

5.3.6 向下扩展的标注

之前例子中所做的标注都属于第四章开始处提到的向上扩展的标注，也即我们先对一个内部（或者层次更加深）的物体进行标注，然后通过规则获取层次结构中上层的物体标注。例如在之前的例子中，会议室包含了会议桌和座椅，因此将后者作为部分标注来获取前者的标注是一种向上扩展的标注。利用本文所提出的规则的灵活性，我们还可以实现向下扩展的标注，即先对外部（或者层次较浅）的物体进行标注，然后利用规则获取其内部物体的语义信息。作为这个过程的演示，我们可以将 5.3.2 节中的规则第一条改为如下的文本：

```
((and    (instance-of y Object)
         (instance-of x MeetingRoom)
         (y at-center x))
 ((instance-of y MeetingTable))
```

随后，我们如图 5-7 一样对场景进行部分标注，这一次，我们将对会议室本身进行标注，也即将它所代表的 X3D 节点和会议室的本体实例进行关联。最后，我们启动自动标注系统，该系统将给出如图 5-13 所示的标注内容。该图中所示的标注信息与图 5-8，

图 5-9 中所示的大致相同。区别在于先前我们对一个座椅进行了标注，因此该标注本身将不出现在自动生成的标注中，通过序号可以发现，图 5-13 中所示的关于座椅的标注比之前多了一条。

Suggested Annotations

Subject	Property	Value
instance_138:MeetingChair	in-room	instance_136:MeetingRoom
instance_138:MeetingChair	has-xml-node	meetingroom.x3d/x3d/Transform[4]
instance_139:MeetingChair	in-room	instance_136:MeetingRoom
instance_139:MeetingChair	has-xml-node	meetingroom.x3d/x3d/Transform[5]
instance_140:MeetingChair	in-room	instance_136:MeetingRoom
instance_140:MeetingChair	has-xml-node	meetingroom.x3d/x3d/Transform[6]
.....		
instance_149:MeetingChair	in-room	instance_136:MeetingRoom
instance_149:MeetingChair	has-xml-node	meetingroom.x3d/x3d/Transform[15]
instance_137:MeetingTable	has-xml-node	meetingroom.x3d/x3d/Transform[20],
instance_137:MeetingTable	in-room	instance_116:MeetingRoom
instance_136:MeetingRoom	has-table	instance_117:MeetingTable
instance_136:MeetingRoom	has-chairs	12:Integer

图 5-13 用向下扩展的方式得到的标注内容

Fig.5-13 Annotation Content Obtained via Downward-Extension Annotation

此外，我们还可以通过修改规则，来调整一个会议室能够被称之为大型的条件要求。例如对于 5.3.2 小节中提出的规则，我们可以将规则 2 中的(greater-than num 10)一项改成其他的计算公式，来改变会议室规模的判断方式。

5.4 结果总结与讨论

5.4.1 研究结果的应用意义

上述的例子实现了从本体构建，规则编辑，部分标注，到最后自动生成标注内容的完整流程，并且其最后的标注内容和最初预期的一致。该结果验证了我们的标注引擎可以以层级的方式来生成标注，也即通过标注引擎生成的标注内容本身可以被看做部分标注来产生新的标注内容。因此，标注者在编辑标注规则时也可以将层级的标注思想体现在标注规则内，从而最大程度地减少他们最初所需要做的初始标注。

另外从整体上来讲，该结果也意味着当一组三维场景的标注方式存在许多共性时，标注人员可以用本文提出的框架快速地搭建自动标注系统，来达到高效地对它们进行标注的目标。当这一目标达成时，也就意味着我们有能力去实现本文第二章开头所提出的

应用软件系统。这些软件系统将利用标注系统所生成的标注内容作为知识库，来对场景的特征和场景内部的结构特征进行精确的检索。以第二章引言部分的家居装潢电子商务系统为例，当我们对不同的居室进行标注完毕以后，该场景的提供商就能够搭建软件系统来为用户提供查询接口。通过这些接口，最终用户可以准确地查询拥有特定属性值（例如楼层，采光等条件）的居室，或者根据居室间的相互连接关系查询到拥有某种特定居室布局的房型。

5.4.2 系统的不足之处

在有些情况下，标注引擎可能无法得到完整的结果。这其中最主要的情况是规则条件涉及到复杂的空间关系时。以上述例子来讲，我们要求会议桌相对于房间的位置为中央位置。但中央位置这个概念的语义却是较为模糊的。我们在第三章的表 3-3 中定义了若干空间属性的数学定义，在这些定义中都利用到了阈值或者可配置的参数。我们也在本章的系统实现中介绍了如何用接口的方式允许同一个属性多种抽取策略可替换的机制。因此，同一种空间属性并非具有严格定义。在不同的计算方式下，某个会议桌与其所在房间的空间关系可能会返回不同的结果。这是由三维场景基于内容的分析所具有的不严格性本身所决定的，因此对于某一个特定的领域，若期望得到更好的标注结果，应当根据所在的领域特别订制个别可视化信息抽取器的实现，使得它们的行为更符合当前领域的特征。

另一方面，利用半自动化标注引擎所得到的标注效率提升的程度，和规则的利用次数成正比，也和部分标注的质量存在一定关系。在上述例子中，我们利用到所有的座椅都是由同一模型复制而成这一特点。如果它们不是以这种方式在场景中被摆放，或者该场景中所有的座椅都为不同样式，则要使得上一节中的标注规则得以应用，我们还必须部分标注所有房间内的座椅。这种情况下，我们能够得到的效率提升就会较之之前有所下降。

5.4.3 与现有标注系统的比较

表 5-1 列举了从准确性，自动性等方面对本文提出的研究，和另外两种相关的研究进行了对比。比较的对象分别是 ShapeAnnotator[19]和 ToBoA[27]，它们都是基于本体针对三维场景进行语义标注的方法。在表中可以看出，ToBoA 由于采用了手动的标注而具有较高的准确性，包括本文在内的其他两种标注手段则因为引入了半自动特性而牺牲了部分准确性。另外，本文和 ToBoA 都利用了三维场景描述中自顶向下的特征，因此对于物体的形状特征把握略显欠缺。

表 5-1 三种基于本体的三维场景语义标注方法的比较

Table5-1 Comparison between 3 ontology-based methods of annotation on 3D scenes

特征	标注方法		
	本文	ShapeAnnotator	ToBoA
自动性	半自动	半自动	手动
准确性	中等	中等	高
三维内容	自顶向下的层次结构	网格结构	自顶向下的层次结构
本体	通用本体，领域本体的多方面本体	领域本体	层次化的领域本体
物体关系的抽取	高	中等	中等
查询接口	本体描述	本体描述	本体描述+关键字
自动化学习方式	基于用户自定义规则	基于系统预定义特征描述符	无
对领域专家的依赖	强	强	中等

本文相对于现有研究的优势是：

（1）本文利用三维文档中自顶向下的描述格式来抽取物体自身的几何属性以及物体间的空间关系。这些信息帮助了标注人员定义标注规则，并且使得最终用户可以对场景内部的物体结构及属性进行检索。本文在第三章讨论了可视化信息的抽取，在 5.3.5 节展示了该特点在原型系统中的体现。

（2）本文定义了灵活的规则来辅助标注人员抽象重复的或者类似的标注过程。这提高了标注工作的效率，也间接降低了语义标注工程的成本，使得三维模型或场景供应商能够更快地应对更大量文档急需标注的需求，及时地为他们的最终用户提供良好的查询及检索服务。本文在第四章讨论了标注规则的编写，匹配和实施，在 5.3.6 小节展示了用不同的规则达到不同效果的应用场景。

5.5 本章小结

本章具体介绍面向 X3D 基于本体的自动语义标注系统的实现和验证。本章首先给出原型系统的总体架构，随后对其中的部分模块的实现方式进行阐述。本章的后半部分通过一个具体的应用实例来介绍如何使用我们的系统来完成一个完整的标注流程，并由应用的结果来验证本文所提出方法的实用性和有效性。该有效性包括正确地抽取出第三章介绍的可视化信息，以及利用规则的灵活性达到向上和向下扩展标注的能力。本章最后将本文和其余两种相关的研究就若干方面进行了比较和讨论。

6 总结与展望

6.1 主要研究工作总结

本文提出了一套基于本体面向 X3D 三维场景文档的半自动语义标注框架，旨在为三维文档的快速语义标注提供一个可行的解决方案。本文提出的框架利用本体作为语义标注的知识表现载体，利用规则定义来实现软件执行的自动化。本文的主要工作包括以下几点：

其一，提出基于本体面向 X3D 文档的半自动语义标注框架的总体架构，定义该框架与外界的交互行为以及其内部的主要功能模块；确定一个典型的标注流程在该框架中执行的主要步骤。

其二，对基于三维文档的语义标注进行本体建模。提出用于描述三维场景建模中基础要素的通用本体概念，给出通用本体需要满足的条件以及一个典型的通用本体实现。提出用于标注文档结构的文档本体和实现。提出利用通用本体进行领域本体的建模。

其三，提出从三维场景文档中抽取可视化信息的算法。这包括，识别出三维文档中所包含的可视化信息的种类；为每一种可视化概念类提供抽取算法；为每一种属性建立计算方法；为每一种可合并的属性建立合并的算法；为牵涉到多个物体的空间属性提供计算公式。

其四，建立根据用户制定的标注规则执行自动标注的标注引擎。这个过程首先包括提出用于描述自动标注行为的标注规则；给出标注规则的一般语法，规则定义中可以用到的标识符种类；定义用简单规则语法创建复杂的规则描述的方法。在此基础上，给出了标注规则一致性检测的算法，分别针对单个标注内部的一致性以及多个标注循环触发的可能性给出检测策略。随后，提出能够在大量三维场景文档中迅速找出符合条件的标注规则和场景节点，并给出从标注规则文本到该数据结构的构造方式，以及如何将抽取出的可视化信息分配到该数据结构中启动标注引擎的算法。

最后，建立原型系统。将本文提出的思想及算法实现为一个可运行的软件系统。该系统作为一个标注系统，包含以下若干功能：导入 X3D 场景文件并从中抽取出可视化信息；导入通用本体，文档本体以及领域本体，指导用户编写标注规则；显示 X3D 文件渲染场景及源代码，辅助用户进行部分标注的完成；导入待标注三维文档，各类本体以及部分标注，进行自动标注，并将标注内容进行持久化。作为系统的验证，选取了办

公楼建模作为应用场景，建立了三维场景，标注规则，领域本体以及部分标注，验证了由标注引擎所自动生成的标注内容的正确性。

6.2 未来的工作展望

面向 X3D 基于本体的半自动化语义标注方法，还有如下问题有待进一步的研究：

第一，在本文现有的研究中，物体的形状信息难以计算。实际上，较多的标注行为都可以以物体的形状进行描述，在未来的研究中需要将物体的形状抽取融入到可视化信息抽取过程中去。

第二，面对大量规则或场景文档时，本文提出的规则匹配和应用算法仍然是不够的，因为需要建立庞大的条件网络图。在未来的研究中，将着眼于如何在该算法上进行优化。

第三，本文提出的方法仍然处于原型系统的阶段，面对企业级的使用仍然有诸多可扩展性和健壮性的问题需要考虑。

参考文献

[1] Berners-Lee, T., Hendler, J., & Lassila, O. (2001). The semantic web. Scientific american, 284(5), 28-37.

[2] Petridis, K., Anastasopoulos, D., Saathoff, C., Timmermann, N., Kompatsiaris, Y., & Staab, S. (2006). M-ontoMat-annotizer: Image annotation linking ontologies and multimedia low-level features. In Knowledge-Based Intelligent Information and Engineering Systems (pp. 633-640). Springer Berlin/Heidelberg.

[3] Tang, Jie, et al. "Automatic Semantic Annotation Using Machine Learning." The Semantic Web for Knowledge and Data Management: Technologies and Practices (2009).

[4] Gao, B., Zheng, H., & Zhang, S. (2009, November). An overview of semantics processing in content-based 3d model retrieval. In Artificial Intelligence and Computational Intelligence, 2009. AICI'09. International Conference on (Vol. 2, pp. 54-59). IEEE.

[5] Xu, D., & Li, H. (2007, August). 3D shape retrieval integrated with classification information. In Image and Graphics, 2007. ICIG 2007. Fourth International Conference on (pp. 774-779). IEEE.

[6] Princeton Shape Benchmark. http://shape.cs.princeton.edu/benchmark/

[7] Ohbuchi, R., Yamamoto, A., & Kobayashi, J. (2007, September). Learning semantic categories for 3D model retrieval. In Proceedings of the international workshop on Workshop on multimedia information retrieval (pp. 31-40). ACM.

[8] Klien, E. (2007). A Rule‐Based Strategy for the Semantic Annotation of Geodata. Transactions in GIS, 11(3), 437-452.

[9] Djioua, B., Flores, J. G., Blais, A., Desclés, J. P., Guibert, G., Jackiewicz, A., Priol, F., Nait-Baha, L., Sauzay, B. (2006). EXCOM: an automatic annotation engine for semantic information. FLAIRS 2006, 285-290.

[10] Flickr，www.flickr.com/

[11] Stackoverflow，stackoverflow.com/

[12] Kiryakov, A., Popov, B., Terziev, I., Manov, D., & Ognyanoff, D. (2004). Semantic annotation, indexing, and retrieval. Web Semantics: Science, Services and Agents on the World Wide Web, 2(1), 49-79.

[13] Hollink, L., Schreiber, G., Wielemaker, J., & Wielinga, B. (2003, October). Semantic

annotation of image collections. In Knowledge Capture (pp. 41-48).
[14] Assfalg, J., Bertini, M., Colombo, C., Del Bimbo, A., & Nunziati, W. (2003). Semantic annotation of soccer videos: automatic highlights identification. Computer Vision and Image Understanding, 92(2), 285-305.
[15] Bloehdorn, S., Petridis, K., Saathoff, C., Simou, N., Tzouvaras, V., Avrithis, Y., Handschuh, S., Kompatsiaris, Y., Staab, S., Strintzis, M. (2005). Semantic annotation of images and videos for multimedia analysis. The semantic web: research and applications, 592-607.
[16] 潘翔, 张三元, & 叶修梓. (2009). 三维模型语义检索研究进展 [J]. 计算机学报, 32(6), 1069-1079.
[17] 杨育彬, 林珲, & 朱庆. (2004). 基于内容的三维模型检索综述. 计算机学报, 27(010), 1297-1310.
[18] Attene, M., Robbiano, F., Spagnuolo, M., & Falcidieno, B. (2009). Characterization of 3D shape parts for semantic annotation. Computer-Aided Design, 41(10), 756-763.
[19] 王新颖, 王生生, 吕天阳, 庞纪明, & 王钲旋. (2007). Onto3D——一种基于本体的三维模型检索系统. 计算机研究与发展, 2.
[20] X3D Specs. http://www.web3d.org/x3d/specifications/x3d_specification.html
[21] 林冬梅. (2004). X3D: 新一代 WEB 交互式三维图形规范. 计算机系统应用, 10, 1038-1043.
[22] X3D. http://en.wikipedia.org/wiki/X3D
[23] 曹雪. (2011). 面向网络协同的语义驱动的三维场景设计复用研究 (Master's thesis, 上海交通大学).
[24] Pittarello, F., & De Faveri, A. (2006, April). Semantic description of 3D environments: a proposal based on web standards. In Proceedings of the eleventh international conference on 3D web technology (pp. 85-95). ACM.
[25] Pittarello, F., & Gatto, I. (2012, May). A visual interface for querying ontologically and socially annotated 3D worlds for the web. In Proceedings of the International Working Conference on Advanced Visual Interfaces (pp. 377-381). ACM.
[26] Pittarello, F., & Gatto, I. (2012, May). An integrated ecosystem of interfaces for annotating, querying and browsing networks of Web3D worlds. In Proceedings of the International Working Conference on Advanced Visual Interfaces (pp. 794-795). ACM.
[27] Pittarello, F. (2009). An approach for tagging 3D worlds for the net. CELENTANO A.; YOSHITAKA A.; MARESCA P.; JIN Q, 88-93.
[28] Pittarello, F., & Gatto, I. (2011, June). ToBoA-3D: an architecture for managing

top-down and bottom-up annotated 3D objects and spaces on the web. In Proceedings of the 16th International Conference on 3D Web Technology (pp. 57-65). ACM.
[29] OWL Web Ontology Language. http://www.w3.org/TR/owl-features/
[30] 张金钊, 张金锐, & 张金镝. (2007). X3D 虚拟现实设计: 第二代立体网络程序语言. 电子工业出版社.
[31] 张金钊, 张金锐, & 张金镝. (2008). X3D 立体动画与游戏设计: 第二代立体网络程序设计实例. 电子工业出版社.
[32] ZHANG, Y., & XIA, Q. (2006). JAVA Rules Engines Based on Rete Algorithm. Science Technology and Engineering, 11, 021.
[33] JDOM. www.jdom.org/
[34] Jena. jena.apache.org/
[35] SPARQL. www.w3.org/TR/rdf-sparql-query/
[36] xj3d. http://www.xj3d.org/
[37] 李贞培, 李平, & 吴明. (2010). X3D 虚拟场景与 Java 交互研究. 计算机工程与应用, 46(016), 67-70.
[38] The Protégé Ontology Editor. protege.stanford.edu/

致 谢

值此硕士毕业论文完稿之极，我谨向所有在过去二余载中帮助，鼓励过我的人表示衷心的感谢！

首先，我要感谢我的导师蔡鸿明老师和姜丽红老师，感谢你们在研究思路和学习方法上对我的悉心指导。蔡老师在为我提供了宽松的学习环境的同时，也在研究方向上给予了我重要的意见，指引我完成最终的研究成果。姜老师为学严谨，常常能提出尖锐的问题，帮助我修正并完善研究中的瑕疵之处。

其次，我要感谢研究生阶段在生活上学习上帮助过我的同学和朋友们。这其中，尤其要感谢实验室内谢诚，黄承曦两位博士，每次与你们讨论问题都能得到许多启发；感谢曹雪，汪涵两位可视化小组的成员，没有你们打下的基础，将不会有本文的研究成果；感谢朋友非凡，在我遇到逆境和不顺时开导并鼓励我。

再次，我要感谢软件学院为学生提供的硬件设施，帮助我们在优良的实验室环境中学习和生活；感谢上海交通大学提供的优秀的网络资源和完善的学术资源，保障我的研究顺利进行。

最后，我要感谢一直以来默默支持着我的家人们。近三年来，无论我遇到什么样的挫折和失败，你们都会在第一时间关心我，为我出谋划策。感谢父亲在健康调理上对我的帮助，感谢母亲在生活自理上对我的关心。

攻读学位期间发表的学术论文目录

[1] Mengwei Shi, Hongming Cai, Lihong Jiang，An Approach to Semi-Automatic Semantic Annotation on Web3D Scenes Based on An Ontology Framework. In 2012 The International Conference on Intelligent Systems Design and Applications (ISDA)