中图分类号：O123, O244, TP399

论 文 编 号 ：10006BY0409103

北京航空航天大學

博士学位论文

电子几何教科书系统的设计与实施

作者姓名　　陈肖宇

学科专业　　基础数学

指导教师　　王东明　教授

培养学院　　数学与系统科学学院

On the Design and Implementation of an Electronic Geometry Textbook System

A Dissertation Submitted for the Degree of Doctor of Philosophy

Candidate: Chen Xiaoyu

Supervisor: Prof. Wang Dongming

School of Mathematics and Systems Science

Beihang University, Beijing, China

中图分类号：O123, O244, TP399

论 文 编 号 ：10006BY0409103

博 士 学 位 论 文

电子几何教科书系统的设计与实施

作者姓名　　陈 肖 宇　　申请学位级别　博 士

指导教师姓名　王 东 明　　职　　称　　教 授

学科专业　　基础数学　　研究方向　　数学知识管理

学习时间自　　年　　月　　日　　起至　　年　　月　　日止

论文提交日期　　年　　月　　日　论文答辩日期　　年　　月　　日

学位授予单位　北京航空航天大学　　学位授予日期　　年　　月　　日

关于学位论文的独创性声明

本人郑重声明：所呈交的论文是本人在指导教师指导下独立进行研究工作所取得的成果，论文中有关资料和数据是实事求是的。尽我所知，除文中已经加以标注和致谢外，本论文不包含其他人已经发表或撰写的研究成果，也不包含本人或他人为获得北京航空航天大学或其它教育机构的学位或学历证书而使用过的材料。与我一同工作的同志对研究所做的任何贡献均已在论文中作出了明确的说明。

若有不实之处，本人愿意承担相关法律责任。

学位论文作者签名：_____ 日期：　　年　　月　　日

学位论文使用授权书

本人完全同意北京航空航天大学有权使用本学位论文（包括但不限于其印刷版和电子版），使用方式包括但不限于：保留学位论文，按规定向国家有关部门（机构）送交学位论文，以学术交流为目的赠送和交换学位论文，允许学位论文被查阅、借阅和复印，将学位论文的全部或部分内容编入有关数据库进行检索，采用影印、缩印或其他复制手段保存学位论文。

保密学位论文在解密后的使用授权同上。

学位论文作者签名：_____ 日期：　　年　　月　　日

指导教师签名：_____ 日期：　　年　　月　　日

摘　要

数学知识管理是近十几年来兴起的一门交叉学科, 主要研究如何利用先进的计算机科学与技术, 提出新的思路和方法, 设计开发完善的软件工具来更加有效地管理数学知识, 以满足人们对数学知识的创造、发布、共享、访问及使用等需求. 作为历史悠久的数学分支, 几何学包含内容丰富且形式多样的几何知识. 要对其进行有效管理, 我们需要根据其特点研究专门的管理方法. 本文参考一般数学知识管理的研究思想和方法, 基于近年来几何定理机械化证明技术以及动态几何软件的研发状况, 提出了一个新的研究方向 —— 几何知识管理, 阐述了其研究的主要内容和面临的挑战, 并给出了一套获取、表示、封装、分类、组织、维护、呈现、转化、处理几何知识的管理方法和技术.

应用这些方法和技术, 我们从知识管理的角度, 针对目前以静态文档模式表示知识的传统教科书的不足, 提出、设计并实现了一个以动态教科书的形式来表示并管理 (欧几里德平面) 几何知识的系统 —— 电子几何教科书系统. 该系统可以辅助用户细粒度地创建、维护、共享、浏览包含动态几何图形以及交叉引用的不同版本的几何教科书或文档; 实时地自动检测所构建的教科书叙述结构是否合理, 内容是否全面以及是否出现重复; 通过调用外部几何软件, 自动证明教科书所包含的命题, 自动构造并绘制相应的动态几何图形. 在实现这些功能的过程中, 我们研究并解决了几何知识管理的一些主要问题.

通过整理和分析几何知识的特征, 我们设计实现了一个几何知识库, 用于存储和管理多种版本不同形式的几何知识数据以及刻画它们之间关系和分类体系的元知识数据, 并构建了近千条几何定义、定理、证明等知识数据. 这个知识库可服务于几何自动推理与计算、自动作图、自动翻译、同义转化、不同版本的文档生成等应用.

通过应用知识库中的元知识数据, 我们研究、设计并实现了实时检测教科书叙述结构一致性、内容完备性和冗余性的算法, 从而有效地辅助教科书合理地构建和维护.

我们设计了一种易于使用的形式化几何描述语言, 并以此作为中间语言, 研究解决了几何表述自动处理的相关问题, 包括几何文档中概念的精确匹配、几何知识表述之间 (特定) 关系的自动发现、几何表述的同义转化 (应用概念的定义), 进而实现了这种语言与外部几何定理证明器 (GEOTHER) 和动态几何软件 (GeoGebra) 的接口. 所提出的几

何描述语言以及相关方法有效地辅助了教科书内容的正确性检测以及几何知识的元知识构建.

关键词: 几何表述语言, 同义转化, 动态教科书, 几何知识库, 几何知识管理

Abstract

Mathematical Knowledge Management is a new emerging interdisciplinary field of research. Its objective is making full use of computer science and technology to propose new methodologies and develop sophisticated software tools for efficiently creating, disseminating, sharing, accessing, and processing mathematical knowledge. As one of the oldest mathematics subjects, geometry embraces a large amount of knowledge represented in different forms. Specialized methods taking into account the features of geometric knowledge are needed for efficient management of the knowledge. Referring to general research ideas and methodologies of Mathematical Knowledge Management and based on the current state of mechanical geometry theorem proving techniques and dynamic geometry software development, this thesis presents a new research direction — Geometric Knowledge Management, explains its main topics and research challenges, and introduces a series of management methods and techniques for geometric knowledge acquiring, representation, encapsulation, classification, organization, maintaining, rendering, transformation, and processing.

Applying these methods and techniques and aiming to improve disadvantages of traditional textbooks in the form of static documents, we propose, design, and implement a system for representing and managing (plane Euclidean) geometric knowledge in the form of dynamic software. This system, called an *Electronic Geometry Textbook*, can assist users to construct, maintain, share, and browse customized geometry textbooks or documents at a detailed granularity, is capable of automatically checking whether the presentation structure of the constructed textbook is appropriate for users to read and learn, and whether the contents are complete or redundant. It also interfaces with external geometry software systems for automatically proving theorems therein and drawing their dynamic diagrams. To implement these functions, we study some main problems on Geometric Knowledge Management.

By investigating and analyzing the characteristics of geometric knowledge, we

design and implement a knowledge base that stores and manages multiversion geometric knowledge data for different applications, together with metaknowledge data characterizing their relationships and taxonomic hierarchies. We construct knowledge data for nearly one thousand definitions, theorems, and proofs. This knowledge base serves for applications in automated geometry deduction and computation, automated diagram construction, automated translation, equivalent transformation, and document generation in different versions.

By applying the metaknowledge data stored in the knowledge base, we implement algorithms for checking the consistency of the presentation structure and the completeness and redundancy of the contents of each constructed textbook in real time, so that the system is able to assist users to create and maintain appropriate textbooks efficiently.

We design a user-friendly formal Geometry Description Language. Adopting it as an intermediate format, we address some problems on automated processing of geometric statements in traditional geometry documents, including precise matching of concepts, automatic discovery of certain relations amongst geometric knowledge statements, and equivalent transformation of these statements (by applying the stored definitions), and we also implement interfaces with an external geometry theorem prover (GEOTHER) and dynamic geometry software (GeoGebra). This language and related methods efficiently support automatically verifying textbook contents and creating metaknowledge of geometric knowledge.

Key words: Geometric formulation language, Equivalent transformation, Dynamic textbook, Geometric knowledge base, Geometric knowledge management

目录

第一章 绪 论

1.1 数学知识管理简介

数学这一历史悠久的学科, 一直以来对科学、技术、工程等各个领域的发展都起着重大的推动作用. 通过数百年数学工作者的思考、总结与积累, 尤其是近几十年以来, 由于大量的人力、财力、时间被投入到新数学结果的探索发现之中, 数学领域不论从深度还是广度上都有了空前的发展, 数学知识的规模不断扩大, 数学分支也纷繁复杂. 如何有效地管理已知的数学知识也随之成为一个需要研究和解决的问题. 不可否认, 相比于几十年前传统的人工信息管理方式, 如今先进的计算机技术, 尤其是因特网以及无线通信设备的普及和使用, 使得信息的传递、共享、交流变得越来越容易, 也使得人们可以方便、快捷、高效地跨越时间的界限与空间的阻隔, 访问规模庞大的数字资源, 从中获取有用的信息, 并通过对这些信息的重新加工和再创造来扩充已有的资源. 特别地, 数学工作者针对所考察的问题, 定义新的数学概念, 通过逻辑推理发现其蕴含的性质、特征、用途, 并得到相关的定理以及理论, 然后将有意义的结果发布在期刊、杂志、图书、报告、会议录等文献中形成数学知识. 其它数学工作者通过查询工具访问获取相关文献并在已有知识的基础上提出新的问题进而重复上一过程. 数学知识正是在这种循环往复的过程中得到不断的积累和扩充. 信息技术的应用使得知识的产生、流通、利用变得更加容易, 也正在改变着人们对数学知识本身的认识. 然而, 我们的努力还远没有达到理想的效果, 数学知识仍然缺少有效的管理.

虽然目前数学知识大部分已经被数字化并存储在各种电子文献数据库中, 人们也建立了多种索引以便更加准确快速地进行查阅, 但数学知识仍然以人可理解而非计算机可理解的格式被封装在独立的文档中, 大多数的数据库索引方法都是通过关键字匹配的模式来实现检索需求的, 例如匹配标题、作者、主题词等文档元数据, 或者文档中重复出现的关键字、短语等字段. 这种查询知识的方式仍然停留在语义处理的较低层次上, 以至于所查结果或者由于粒度太粗糙而使真正有用的知识埋没于大量其它没多大关系的知识中, 或者难于准确地满足人们的使用需求, 因而延缓了知识流通的速度, 影响了知识的重新使用和新知识的发现. 目前大量产生于实践当中, 尤其在工程领域使用的与算法有关的数学知识, 并没有被有效地获取及数字化, 因而不能被传播、共享和应用. 有时由

于没有一种有效的手段来获得全面和完备的知识, 以至于投入精力和时间思考研究得到的结果却是他人已经做过的, 甚至可能出现相同的问题被不同个体重复地解决许多次的情况. 对于非某领域的数学工作者或者工程领域的工作者而言, 要获知此领域已经存在哪些数学知识, 这些知识之间有哪些联系以及哪些知识可以应用到具体的工程问题当中并不容易, 只有通过把搜集的相关文献阅读理解之后才能了解理论的层次结构, 明确知识的来龙去脉, 进而才能准确地获得对自己有价值的知识并做出应用和创新. 不同个体对知识的整理以及结构的重新认识浪费了人们的时间和精力, 降低了知识的使用效率.

另一方面, 目前, 数学工作者做数学的方式仍然主要依靠自身的思考, 即知识的发现、审核、评价、应用等都是通过人来完成的. 随着计算机技术的发展, 人们开发了大量用于解决数学问题的软件系统, 尤其是用于自动推理与证明, 证明构造与验证, 符号与数值计算等方面的工具使得计算机能够象人一样进行逻辑推理和计算. 这些工具的使用可以部分地代替人们的思考, 节省人们的时间与精力, 在知识的发现与创新, 评审与发布, 传播与教学等方面都必将起到辅助作用, 从而提高效率. 然而这些工具往往以大量形式化的知识数据作为操作对象. 由于目前大部分的数学知识是以非计算机可理解的格式进行存储和管理, 并且应该通过何种方式来有效地利用这些工具仍有待考察, 因此它们所发挥的作用受到了很大的限制.

管理数学知识的这些传统方式已经不足以满足需求, 我们需要探索新观念、新方法、新理论、新技术来重新审视数学与数学知识.

近十几年兴起了一个由数学、计算机科学、图书馆学和出版业相交叉的研究领域 —— 数学知识管理 (Mathematical Knowledge Management), 主要研究的目标是应用先进的计算机科学与技术探索新方法, 开辟新思路, 设计开发完善的软件工具来更加有效地管理 —— 表示、组织、发布、访问以及处理数学知识, 使计算机能理解数学知识的语义并进行必要的推理、计算、转化、检索, 以帮助人们摆脱在数学研究过程中平凡甚至机械的劳动而将精力集中于创造性的思考上. 数学知识管理将服务于发现及使用数学知识的数学工作者、科学家、工程师, 传授数学知识的老师, 学习数学知识的学生, 提供数学教科书等并发布数学结果的出版者, 编目组织数学知识的图书管理工作者等 [19,41]. 数学知识管理与人工智能领域知识工程的区别之处在于它所研究的内容是要满足人们对数学知识的多种层次的需求, 辅助人们来解决数学问题, 而不是模拟人代替人来解决问题.

作为一个研究领域, 数学知识管理是 2001 年 9 月在奥地利哈根伯格举行的第一届数学知识管理国际研讨会 (MKM 2001) 上被明确提出来的 [17]. Buchberger B. 将数学知识管理定义为“数学知识的计算机辅助管理”. 这一会议至今已成功举办了九届. 由于形式化数学知识的方法 (从经典的一阶逻辑到高阶依赖类型的多态 λ 演算) 以及看待数学知识的角度和观念不同, 数学知识管理也包含了许多子领域及其各自研究的基本问题. 在综述 [19] 中, Carette J. 和 Farmer W. M. 分析总结了之前近 10 年数学知识管理领域的文献, 提出了数学知识管理研究的 6 种观点及其涵盖的 25 个课题. 这些课题涉及数学知识的表示、标注、呈现、提取、机械化、搜索、转化、组织、维护、交互、处理及使用等问题. 尽管数学知识管理研究的角度不尽相同, 我们借鉴 Farmer W. M. 在 [41] 中所阐述的对数学知识进行管理的思路, 从数学知识的共享与使用的角度出发, 将数学知识管理研究所涉及的主要内容概括为如下三个方面.

- 标准化: 数学知识需要被识别、获取、检索、发布、共享、验证并被用于推理和计算, 因此只有标准化后才可以进行相互交流和访问处理. 而仅将数学知识表示为计算机可读的文档格式 (如 PDF、PostScript) 是不足以满足这些应用需求的. 针对不同的需求使用合适的语言对所获取的数学知识进行表述并使用易于计算机处理的格式来形成标准化的知识数据是管理的必要条件.
- 结构化: 数学知识的规模是巨大的, 但它们之间并不是相互独立的而是紧密相连的. 数学知识都存在于特定数学理论的语境之中. 因此, 需要以合适的粒度将数学知识个体化, 获取它们之间的关系并按其结构 (包括其内部的数据结构和理论的逻辑结构) 进行分类、组织、规划, 确保知识的完备、逻辑结构的一致, 避免知识的冗余, 从而为各种应用操作提供合理有效的数据支持.
- 访问与交互: 数学知识的使用者包括人和机器. 人们需要软件工具来发布、共享数学知识, 查找、获取、访问所需要的数学知识并浏览其内容和结构; 计算机需要处理并使用数学知识以帮助人们进行自动 (或半自动的) 推理和计算, 验证数学理论的正确性等. 这些工具的支持对数学知识的管理是至关重要的.

下面, 我们来详细描述这三方面的研究进展, 侧重于概括主要的研究成果以及相关的软件系统.

1.1.1 数学知识的标准化

获取

在过去, 数学知识的发现都是从数学自身的角度出发并依赖于数学工作者的研究兴趣, 主要以定义、定理、猜想、证明等声明的形式来表述. 然而进入计算机时代以后的现代数学知识有相当一部分是来自于工程领域, 由非数学工作者从实际应用当中总结提炼而得到的. 其内容和形式与传统的数学知识相比有很大的不同, 然而对解决实际问题却发挥着很重要的作用. 最好的例子便是应用于计算机软件中的数学知识. 软件系统实现的各种算法实质上是以程序的形式来表述的数学知识. 由于观念上的差异, 数学工作者并不关心这些数学知识. 正由于现代数学知识涵盖的范围如此广泛, 所表现的方式如此多样, 我们就更需要扩大获取知识的范围和力度, 不仅从数学本身的领域, 更应该从数学的工程应用领域来进行积累与扩充, 为不同背景的人们提供全面实用的数学知识资源, 并通过现有的技术将其数字化, 使得人们可以方便地共享与访问. 这是一项长期的任务也是一个庞大的工程. 另一方面, 目前大部分数字化资源都以文档为单位, 并以 PDF 等计算机可读格式表示数学知识, 然而嵌入文档中的文本只是普通的字符, 并没有包含关于数学公式、文档结构等的语义信息. 这样的文档只能由人进行阅读和理解, 而计算机却不能识别其中数学表达式的含义. Infty 项目使用光学字符识别技术开发的数学文档阅读系统 InftyReader 可以识别包含数学表达式的文档的扫描图像, 并生成 LaTeX、HTML、XML 等格式表示的文档 [99]. 这种数学文档的反向数字化技术为数学知识的获取提供了有利的工具, 也使得进一步管理现有文献出版物中的数学知识成为可能.

形式化

数学知识需要使用语言进行表达和叙述, 最常见的便是数学教科书等文献中所使用的自然语言. 这种非形式化的语言 (表达上并不精确但易于人们理解) 也部分地具有形式语言 (有精确的语法规则) 的特征, 例如数学理论使用的符号系统以及数学表达式的书写规则, 但这些形式的表述并不能完整全面地刻画数学知识的含义, 就如同一个句子中出现的名词仅是句子的一个组成成分而已并不能表达这个句子的全部含义. 软件工具读取、分析以及处理数学知识, 需要遵循严格的算法来执行相应的程序. 知识的表述与操

作都必须要在严密精确的规则下进行, 因此需要使用形式语言来对这些规则进行约定.

逻辑作为人们推理思维的数学模型, 通过定义精确的语法规则和推理规则实现了对推理过程的模拟与控制, 通过对抽象符号的形式操作实现了对符号所表示的真实世界的理解与认识, 因此为形式化数学知识提供了理论基础. 根据不同的逻辑理论, 数学知识可以被形式化表述成多种逻辑语言, 如理论验证系统 Mizar [97] 使用的经典一阶逻辑, 理论探索系统 Theorema [120] 使用的一阶谓词逻辑, 证明助手 Coq [34] 使用的高阶类型依赖的多态 λ 演算等. 逻辑语言虽然表达能力很强, 但其语法规则的限制使得数学知识的表述复杂而不利于人们的阅读和使用. 为了使得形式化表述的数学知识能够更加自然更贴近日常使用, 人们设计了一些形式语言并研究了相关的证明理论, 如 AutoMath [101]、Mathematical Vernacular [13]、Weak Type Theory [100]. 这些关于数学的形式语言的研究成果使得在计算机上处理与应用数学知识成为可能.

表示

数学知识需要在计算机上表示成某种格式才可以被数学知识管理系统和工具有效地操作和处理, 如一个数学理论中的公理、定义、定理被表示成声明语句; 有些用于描述解决某个问题的方法流程的数学知识需要表示成算法; 有的数学知识可以表示成图表的形式, 如几何学中的几何图形、代数曲线曲面或函数的图象、代数中的交换图等等. 目前数学知识的表示语言因系统而异. 不同的系统依据其使用的形式语言的特点及其功能的实现方式, 制定了各自不同的表示语言. 另外出现了一些独立于系统的表示语言, 即将数学知识看作是一种特殊的信息, 应用信息建模技术, 开发标准化的表示格式来描述数学对象 (数学符号、公式等) 及其语义, 使得数学对象精确化、显示化、通用化. 其中, MathML (Mathematical Markup Language) [93] 的设计目的是为了使数学表达式可以在互联网中被传递、接收以及处理, 就像 HTML 处理文本一样. MathML 是基于 XML 的语言, 包括两类标注: 一类是用于在网页中显示数学内容的 Presentation MathML (可以看成是 HTML 的扩展); 另一类是通过标注数学语义信息使得计算机可以处理数学对象的 Content MathML. MathML 目前可以被大部分浏览器支持 (通过安装必要的插件). 作为另一种基于 XML 的语言, OpenMath [104] 通过内容标注的方法来对数学对象的表示进行标准化 (不考虑其如何显示), 并且通过 Content Dictionaries 来指明 OpenMath 所表达的数学符号的语义 (或含义), 使其可以在相同的语义背景下在不同的计算机程序

或系统中交互使用从而实现计算资源的共享与分布式计算 [64]. 尽管都是对数学表达式的内容进行标注, 两者在语法、精确性以及结构上都存在不同. Content MathML 对数学符号的含义解释只是通过非形式化的自然语言向人进行解释, 而 OpenMath 则是通过提供机器可读的 Content Dictionaries 来定义数学符号的含义.

很多系统都提供了将自身的表示语言转化成 MathML 或 OpenMath 的功能, 如计算机代数系统 Maple 和 Mathematica, 并且这两种语言之间的相互转化以及将两者融合起来的统一格式标准的制定也正在进行当中 [38].

不同的系统根据其内部的表示格式都提供了各自的数学知识创建工具 (authoring tools) 使得编码更加方便和容易. 由于常用的数学表达式通常是以二维的形式 (字符的非线性排列) 出现的, 因此很多 WYSIWYG 工具被开发使得人们可以通过传统的方式来编辑数学表达式而同时得到使用多种表示语言表示的代码, 例如公式编辑器 MathDox formula editor [91]、MathEdit [116] 等通过所见即所得的方式来构建数学公式并得到相应的 MathML、OpenMath 等语言表示的代码, 还有一些工具利用手写识别技术来创建数学表达式 [43].

以上的标准表示语言可以明确无歧义地表示数学对象, 这就使得人与计算机或者不同计算机程序之间在数学对象的语义上达成共识, 但是并没有在数学知识的层次上解决语义表示的问题. 事实上, 数学知识都是在一定的上下文 (语境) 中被理解的, 如数学表达式中所使用的符号或概念需要定义, 定理的存在需要公理假设等前提. 但在通常的数学知识表述中, 这些语境并不会明确地呈现出来, 而需要人们通过阅读来将数学知识贯穿并加以理解来获得. 这样表述的数学知识并不精确或者说并不是可以被计算机所理解的. 因此, 管理数学知识需要明确地表示数学知识的上下文, 即所使用的概念和符号的含义, 逻辑推理过程中所使用的假设前提等, 也只有这样, 软件工具才是可靠的, 才可以有效地分析并处理数学知识. 下面就介绍通过对数学知识进行结构化的方法来解决语境表示的问题.

1.1.2 数学知识的结构化

数学知识并不是孤立的, 而需要置身于特定的数学理论之中, 为其提供语境. 这就需要将所获取的数学知识整理组织在一起并在高于数学知识的层次上来分析建立他们之间的各种关系, 从而清晰明确地表示其结构.

将数学知识结构化的方法很多. 大多数情形下数学知识被组织于如数学教科书等

文档的“章-节-段”结构之中, 并通过交叉引用来刻表示学知识之间的关系. 但这种交叉引用的粒度比较粗糙, 并不能细致完全地刻画数学知识的结构. OMDoc [75] 是一种基于 XML 的标注数学文档的标准语言并可服务于机械化推理系统之间的交互. 它通过如下三个层次对数学知识进行结构化: 对象 (应用 Content MathML 或 OpenMath 表示), 陈述 (如定义、定理、例子、证明等), 理论 (表示不同理论之间的导入、包含和映射等关系). 应用模块化 (Schema) 的思想, OMDoc 对文档内各个部分按照不同层次进行划分标注, 然后通过交叉引用的方式来明确地、细粒度地表示文档的结构. 通过为 OMDoc 文档元素引入相应的类以及这些类的层次结构和性质, 刻画 OMDoc 文档语义的本体也被创建 [80]. MathDox [90] 是一种通过整合多种已有的 XML 格式 (包括 DocBook、OpenMath、MONET、XForms、Jelly 和 XInclude) 来表示和结构化交互式数学文档的标准格式. 这种文档可以被转化为网页并通过在屏幕上的动态演算来演示算法、测试人们的数学技能以及解释数学概念.

很多系统都构建了大规模的数学知识库, 并根据不同的目的使用不同的方法来实现对数学知识的组织和结构化. 例如 ActiveMath [3] 是一个基于 Web 的自适应交互式学习环境, 其学习资源的表示基于 OMDoc 格式, 通过增加教育学相关的属性和关系 (metadata) 的标注来实现其内容的组织与结构化 [96]. 证明助手 Coq 允许用户创建数学定义和断言, 帮助构造和机械化地验证这些断言的形式证明, 并可以从中抽取可信任的计算程序. 用于管理形式化的代数与分析知识的大规模分布式 Coq 知识库 (C-CoRN) 已经被构建 [36]. DLMF (Digital Library of Mathematical Functions) [102] 的目的是构建几十年来一直被工程人员视为标准参考手册的 Handbook of Mathematical Functions [2] 的在线版本以及对其内容进行扩充, 并通过创建 LaTeX 编码的文档, 使用特殊的宏来生成手册的打印版本以及动态的可视化网页 [39,88]. HELM (Hypertextual Electronic Library of Mathematics) [8,9] 的目标是研究一套为建立和维护虚拟的、分布式的、超文本的和形式化的数学知识库而需要的技术, 通过创建 LaTeX 或者 Coq 语言编码的文档, 并将其转化为 Content MathML 表示使得数学知识可以通过网络浏览及搜索. 基于 MathDox 格式表示的微积分和线性代数领域的交互式文档已经被开发和管理并被应用到在线的网络教学与测试中 [4,135]. MBase [77] 是基于 Web 的分布式数学知识库, 构建于 OMDoc 之上, 采用发展图技术对大规模的知识进行结构化. MML (Mizar Mathematical Library) [97] 是当前最大规模的形式化数学知识库, 积累了近千篇可以

被 Mizar 验证器自动验证的 Mizar article (包含了使用 Mizar 语法表达的定义、定理、证明等), 从而实现了数学知识的重新构造 [132]. MOWGLI 系统 [92] 通过整合用于管理和发布数学文档的已有标准格式 (MathML、OpenMath、OMDoc) 来表示和组织来自于 Coq 知识库以及基于 LaTeX 的 Living Reviews in Relativity 期刊文章中的数学知识, 并使用不同的 XML 技术 (XSLT、RDF) 来支持其内容的网络浏览、呈现、索引、在线查阅、搜索及自动 (或机械化) 推理 [10,57]. 国家知识基础设施 (Nation Knowledge Infrastructure, 简称 NKI) [20,142] 是一个庞大的、可共享的知识群体, 它不仅集成了各个学科的公共知识, 而且还融入了专家知识, 这些学科包括医学、军事、物理、数学、信息科学等, 为科研、教学、科普和知识服务提供有效的基础, 使知识共享成为可能. NKI 通过底层的以概念关系模型为理论基础建立起来的各学科的具体知识以及在此基础上各学科的专业知识本体来对知识进行组织和结构化. Theorema 系统 [120] 为数学研究的整个过程 (提出新的概念和猜想, 证明定理, 提取算法, 最后产出新的理论结果) 提供了一个整合的环境, 不仅可以用于编辑数学文档而且可以同时进行计算与证明 (借助于 Mathematica), 并采用标签与命名空间相结合的方式来结构化数学知识 [113].

为了构建结构化的数学知识, 人们设计开发了相关的创建工具并实现了不同表示之间的转化. 例如 Theorema 系统使用的标签管理工具 [107] 使得用户可以显式地将数学内容分为章、节、段, 并且指定内容的类型等信息 (如定义、定理等); ActiveMath 使用的 jEditOQMath [69] 通过整合多个工具来构建并管理基于 OMDoc 的数学课程文档; sTeX 则通过定义大量的语义宏包实现了使用 LaTeX 语法来创建 OMDoc 文档的功能 [71].

构建数学知识库固然需要大量的工作, 然而由于数学知识的规模逐渐扩大, 数学知识库中的数学知识有可能存在着冗余的现象, 即相同的数学知识可能被构建两次. 关于是否应该保持知识库最小性 (即去除冗余) 的问题, [55] 通过对一些情形的讨论提出了这种知识的重复在很多情形下是合理的也是必需的, 这也为数学知识库的构建与维护提供了新的思路.

1.1.3 知识的访问与交互

数学推理与证明

推理一直是数学发展的主要动力, 而定理证明则是数学推理的主要内容. 从毕达哥拉斯、欧几里德到希尔伯特、布尔巴基, 数学证明的本质都被认为是基于公理演绎的推

理, 即从公理系统出发, 按照一定的逻辑推理规则进行有限步操作而得到定理的过程. 数学证明的主要特征包括两方面: 正确性和可理解性. 证明的机械化一直是人们追求的理想. 形式证明理论为这一目标的实现奠定了基础. 形式证明是指用精确无歧义的形式语言书写的证明, 所使用的符号和字母表都有明确的定义, 每一个证明步骤都没有遗漏, 并且都存在机械化的程序来检验其是否符合推理规则, 通过确保每一步的正确性来断定整个证明的正确性. 哥德尔不完备定理指出任何一个形式的系统都存在不可证明的真命题, 因此数学的形式化方法并不是普遍适用的方法, 不能证明所有的真命题. 尽管并不完美, 形式证明依然发挥着重要的作用, 所得的证明是可以被机械化验证甚至自动生成的, 并且正确性是可以被保证的 (在计算机编译的正确性以及运行这一程序的硬件设备均可被信任的意义下). 近二十年里, 人们应用计算机形式地证明了很多基本定理, 如微积分基本定理 (1996)、代数学基本定理 (2000)、四色定理 (2004)、素数定理 (2008) 等 [59].

形式证明虽然保证了推理的严密正确, 但由于其自身过于繁琐, 以至于过程失去了数学意义使人难于理解. 非形式证明是混合自然语言和数学公式来表达的人可以理解的证明, 通常出现在教科书、期刊等文献之中. 相比于形式证明, 一些逻辑推理步骤被省略, 常识性知识或结果被直接用于证明中, 对于几何学等领域, 直观的判定也可以被引入到证明当中, 这些因素使得非形式证明的正确性只能依靠领域专家来判断, 正如目前大量数学论文的结果主要通过审稿专家进行审阅. 理论上, 非形式证明是可以转化为形式证明的, 而实际上, 这几乎不可能实现. 目前, 对于形式证明方法及理论的研究仍然是热点, 重心在于采取折中的方法来研究数学证明, 一方面确保证明的可验证性, 另一方面保持证明的可理解性.

人们开发了很多数学推理系统, 由于推理所使用的方法和理论有不同程度的交叉, 所以不能严格地对这些系统进行分类. 但按照推理过程的侧重不同, 这些系统大致可以分为以下四类:

- 逻辑自动证明系统: 从领域公理出发, 按照推理规则自动生成某一定理的证明, 如 Otter [106];
- 理论探索系统: 将自动证明系统与数学知识库结合起来, 按照自上而下和自下而上的策略, 通过使用知识库中已有的数学知识 (定义、引理、定理、推论等) 来自动发现新的数学知识, 并将其组织到数学知识库中, 生成的证明相比于逻辑自动证明系统更自然, 更简短, 如 HR [33]、MATHsAID [95]、Theorema [16];

- 交互式证明系统: 通过人机协作的方式来构造证明, 并可以确保证明的过程是可靠和正确的, 如 Coq [34]、HOL light [60]、Isabelle [66]、Matita [5]、Ωmega [103];
- 证明/理论验证系统: 验证给定证明或数学理论的正确性或逻辑一致性, 不产生新的证明, 如 Mizar [114]、Scunak [12].

构造性数学与计算

布尔巴基的形式化方法固然重要, 但利用“数学结构”不能统一整个数学, 因为出现越来越多的与计算相关的数学分支难以纳入其中, 例如组合数学. 基于计算的构造性数学随着计算机的出现得到了巨大的发展, 人们开发许多计算机代数系统帮助解决数学计算问题, 如 GAP、Magma、Maple、Mathematica、Singular 等. 与数学计算相关的算法和方法等数学知识对数学的发展发挥着越来越重要的作用. 因而, 数学知识管理的研究还包括将定理自动证明与计算机代数系统结合, 从而更加全面和系统地支持数学研究, 包括创造新的概念和猜想, 使用计算机将猜想转化为定理并尽可能地进行自动证明, 从证明中抽取出算法 (或者至少是部分的), 通过输入数据运行算法产出新的结果, 进一步产生新的猜想并重复上一过程, 例如 Coq [34]、Theorema [16].

数学知识的检索与浏览

数学知识的检索与浏览是数学知识管理的一个重要的研究课题. 目前使用最广泛的检索方式是基于关键词域的条目检索, 例如 MathSciNet、Zentralblatt MATH、MathSearch. 这种检索方式仅使用和数学文档相关联的元信息, 而不考虑数学知识的语义. 为了使数学知识的检索更准确、更全面、更智能, 一种基于数学知识的结构和内容的检索方式被提出并研究, 其过程涉及到文本检索, 模式匹配, 简单的推理, 甚至定理证明. 这种检索方式可以发现数学知识之间的等价关系, 逻辑导出关系等, 对数学知识的结构化以及数学知识库的构建和应用也起着重要的作用 [14]. 数学知识的浏览涉及到数学知识的分类导航, 以及如何通过传统的样式来呈现. 数学知识的检索与浏览工具的开发很大程度上依赖于数学知识的表示格式. 目前大部分已有的数学知识检索方法受到语义网技术的影响, 将数学知识检索看作是信息检索的一个特例, 基于使用语义标注语言表示的数学知识库, 通过开发相应的匹配工具来实现数学知识的检索, 并通过设计不同的样式表来显示数学内容, 例如 ActiveMath [86]、DLMF [141]、Math

Web Search [76,78]、MOWGLI [92]. 此外, 一些系统针对其知识的表示格式提供专门的检索语言和工具, 例如 MML 使用的检索语言 MMLQuery [11] 以及检索工具 Most of Mizar Matches (MoMM), Coq 和 HELM 所使用的基于 metadata 模型的搜索引擎 Whelp [6,7].

1.1.4 意义和重要性

自从上世纪 50 年代以来, 数学知识的容量呈爆炸式增长. 数学知识不仅包含数学工作者的研究成果, 还包含工程技术人员在解决实际问题中总结创造出来的算法和方法等. 数学知识的表示、结构化、检索等并不是新的研究课题, 但随着其规模越来越庞大, 形式越来越多样, 需要研究新的方法, 探索新的途径以期整合多种功能, 全面系统地为人们提供完善的服务, 避免机械的重复劳动. 在计算机和通信系统普遍使用的情况下, 管理数学知识需要新的手段和工具以适应信息技术的转变. 计算机可以帮助人们推理与计算, 网络可以帮助人们发布、访问与获取数学知识. 因此, 设计开发能够创建形式化和结构化的数学知识, 具备浏览和检索功能, 提供推理与计算服务的管理数学知识的系统和工具, 才能满足人们日益增长的对效率的要求. 数学知识管理领域是在信息技术变革的背景下建立起来的, 并且将改变人们对数学与数学知识在传统意义上的观念和认识, 因此对于数学的发展也将产生深远的影响. 感兴趣的读者可参阅 [74] 进一步了解数学知识管理领域的发展现状.

1.2 几何学机械化与动态几何

使用机器代替人类劳动一直是人们千百年来的梦想. 两百多年前的一场工业革命使得人类从繁重的体力劳动中解放出来, 并使人类历史步入一个新时代; 与此同时, 对人类脑力劳动解放的追求仍在进行着. 数学, 作为自然科学的基础, 最纯粹的脑力劳动, 成为许多数学家、哲学家等追求机械化的目标.

数学活动主要包括两种形式: 数学计算和定理证明. 所谓数学机械化, 实质就是以构造性或算法化的方式从事数学研究, 使数学的计算和推理能够按照规定的有限步骤机械地完成. 人们经过千百年的数学研究积累了大量关于数学计算的机械化方法, 例如加减乘除开方运算, 线性代数方程组的 Gauss 消去法等. 然而证明的机械化却一直是一个难题, 根本原因在于证明涉及到许多模糊不清的技巧规律, 诸如经验、知识储备、直觉灵感等.

Euclid 的经典名著《几何原本》提出为几何学建立一套公理系统, 通过应用这些定义、公理进行逻辑演绎来论证命题得到定理, 从而形成一个逻辑严密的几何理论. 这种公理化方法也为现代数学研究乃至科学研究的方法论奠定了基础. 在这种公理化思想的指引下, 数学证明机械化的设想至少可以追溯到 17 世纪的 Leibniz G. W.. 他提出通过数学的方法研究逻辑, 使推理过程更加精确便于计算. 由于当时的社会条件, 这一设想没能实现, 但促进了布尔代数、数理逻辑等学科的研究, 是现代数理逻辑的萌芽. 另一方面, Descartes R. 创造性地将几何学中的“形”与代数学中的“数”这两个在当时被认为是对立的对象统一起来, 创立了解析几何学, 目的在于将几何问题归结为代数问题, 用代数的方法进行计算从而解决几何问题. 尽管这一使数学在思想方法上产生伟大转折的理论没有实现几何学的机械化, 但却为其指出了另一条道路.

吴文俊先生在 [138] 中指出, 数学机械化的真正缔造者是 Hibert. Hilbert 的经典著作《几何基础》为欧几里德几何建立了一套更完备的、独立的、协调的且比较严格的公理系统, 提出了形式公理法, 几何对象本身达到了更高的抽象, 依靠纯粹机械的逻辑演绎来建立几何学理论, 从而避免了欧几里德公理系统演绎推理中的“直观”和“经验”等不严格因素. 尽管这种依靠公理的形式演绎推理方法对所有定理进行机械化证明的构想后来被 Gödel 不完备定理所否定, 但 Hilbert 的努力催生了一门新的学科 —— 数理逻辑. 另一方面, Hilbert 在《几何基础》中展示了如何通过引入某种数系使基于公理的逻辑演绎定理证明转化为基于坐标系统的代数计算定理证明, 并提出了一条定理, 可以对一类被称作纯粹的交点定理 (Hilbert 类) 进行统一地判定, 从而指明了几何定理证明机械化的道路: 要使用一个通用的计算方法来证明几何定理, 定理的证明不是针对某一个定理, 而是针对某一类定理.

上世纪 40 年代计算机的出现极大地推动了几何定理证明机械化的发展. 1950 年 Tarski A. 提出了针对初等几何 (以及初等代数) 这一范围的命题进行机械化判定的一种通用方法, 得到了著名的 Tarski 机械化定理: 通常意义下的初等几何是可以机械化的. 然而, Tarski 的方法注重一般性, 从而导致过程复杂, 效率低下, 以至于到现在也未能实现在计算机上证明有意义的几何定理.

近 30 年来, 关于几何定理机械化证明方法的研究仍在继续, 相关学者提出的大量新方法使得几何定理自动证明成为自动推理领域最活跃的分支. 总体上看, 这些方法仍旧遵循着两条思路: 一条思路是应用公理化方法进行几何定理证明, 可称之为“AI (人工

智能) 方法”; 另一条思路是将几何定理的证明问题转化为某种代数形式描述的问题, 然后通过代数计算以达到证明定理的目的, 可称之为“代数方法”. 接下来, 我们按照这两条思路分别简单地介绍其中比较成功的方法. 鉴于这方面的综述文章 [27,28,125]、专著 [26,29,138] 比较多, 本文对各方法的细节不作精确的介绍, 而着重于方法的基本原理, 有兴趣的读者可参阅文中的具体文献.

1.2.1 机械化证明的理论和方法

坐标代数法

几何定理机械化证明方法取得突破性的进展要归功于吴文俊先生的开创性工作 [136]. 他在中国古代数学高度构造性、算法化和计算性的启发下提出了几何定理机械化证明方法 —— 吴方法. 该方法效率高, 切实可行, 极大地推动了几何学机械化的发展.

吴方法主要分两步. 第一步是几何定理的代数化, 即通过引入某个特征为零的域上的坐标系统, 将所需证明的几何定理中的几何体和几何关系用代数表达式和代数关系式表示. 常用的几何关系对应的代数表达只涉及到多项式的形式. 这样在初等几何意义下的大部分几何定理一般可以表示成形如 $\{H_T, C_T\}$ 的形式, 其中 H_T 是假设部分, 表示一组多项式方程, C_T 是结论部分, 表示一个 (或多个) 多项式方程. 定理证明的问题在数理逻辑上更一般地称为判定问题, 要证明一个几何定理是否成立, 在代数方法下就转化为判定 C_T 是否是 H_T 的逻辑结论的问题, 即寻找假设部分多项式方程组零点集中的某一部分, 使得结论部分多项式方程在这一部分上成立.

吴方法的第二步是在第一步将几何定理代数化的基础上, 按照一定规则使用假设部分的多项式将结论部分多项式中的变元逐个消去 (伪除). 如果得到的结果 (伪余式) 为 0, 则定理成立, 否则再做进一步处理. 这种多项式变元的消去过程完全是算法化的, 可以在计算机上机械地完成.

吴方法可以有效地证明初等几何中的大部分定理 (定理的假设和结论部分代数关系式都可用关于某个域的多项式等式方程来表示), 并且在此过程中能够自动生成一些代数形式表达的附加条件. 定理是在满足这些附加条件的前提下成立的. 通常的几何体和几何关系在公理和定理中的描述都隐含地假设它们是处于正常的一般情形, 而不是某种特殊的退化情形, 例如当说两条直线平行时, 就隐含着它们不会重合为一条直线; 当说一个三角形时, 就隐含着三个顶点不会共线等. 而应用吴方法得到的这些附加条件中一部分

正反映了原几何定理的非退化条件, 即几何定理是在何种意义下成立; 另一部分排除了几何定理在模棱两可的表述下使得定理不成立的那些情形[注1]. [26,127] 介绍了将前一部分代数形式表达的附加条件自动翻译成几何意义上非退化条件的方法.

除了可以证明几何定理, 吴方法还可以用于自动发现未知的几何关系, 自动发现新的几何定理, 自动推导几何轨迹等. 很多学者了解到了吴的工作后, 纷纷学习并研究改进吴方法, 因而出现了许多基于该方法的改进方法 [73,124,130,131]. 吴方法这种基于多项式的消元来证明几何定理的思想引起了自动推理领域学者的极大兴趣, 很多研究者尝试着把已有的代数方法应用到几何定理机械化证明中, 并开始探索新的机械化方法, 例如 Gröbner 基方法 [79]、结式方法 [140] 以及对含有不等式的几何定理进行证明的特定方法 [139] 等.

以上方法所讨论的几何都是在吴意义下的初等几何 (即通过引入数域和坐标系统代数化后的几何, 而且并没有引入微分运算), 通过对以坐标为变元的多项式的计算达到证明几何定理的目的, 因此称为坐标代数法. 总体来说, 这类方法通过将几何问题定性的困难转化为代数问题定量的复杂, 有效地将几何定理证明机械化推向成功, 也是最为强大最有效率的方法. 但是, 由于证明的过程已经转化为关于坐标变元的代数计算, 证明丧失了几何意义, 对人们来说是一个黑箱. 人们只能得到命题成立与否的判定结果, 而无法从几何的角度理解证明. 在吴方法之后, 一些学者从上个世纪 80 年代就试图设计基于矢量计算的机械化证明方法, 以期可以生成漂亮的, 简单的甚至是可读的证明, 并取得了成功. 这类方法的证明过程也是基于某种代数计算, 但不是关于坐标变元而是关于一些有几何意义的量, 如距离、面积、体积、向量、角等, 因此可以称它们为非坐标代数法.

非坐标代数法

应用面积消点来证明几何定理的方法 (称为面积法) 是由张、周、高等学者提出的 [29,143]. 该方法的主要思想是使用平行线段的比率、矢量面积和 Pythagorean 差这三个基本几何量代替坐标, 建立一些基本命题来描述几何体或几何关系与这些几何量之间的相互关系, 并将这些命题作为推理的依据. 证明几何定理的基本过程使用这些基本命题将定理的假设部分和结论部分分别表示成关于基本几何量的等式关系, 然后机械化

[注1] 通常这些定理在几何构型上会出现多种情形, 如两圆相切包含内切和外切两种情形, 而且一般可使用代数不等式对它们进行区分.

地用假设部分所包含的等式将结论部分等式中的点依次消去 (因为基本几何量是使用点作为基本单元来表示的, 消去的过程事实上就是等量替换的过程), 如果最后能得到一个恒等式, 那么定理证明成功.

面积法的最大优点是证明的每一步都有明确的几何意义, 是第一个强大且高效地自动生成可读证明的几何定理机械化证明方法. 应用面积法人们已经成功地自动证明了 500 多个不平凡的几何定理, 甚至有些证明比几何学家给出的证明还要简短 [44], 但方法的适用范围没有坐标代数法那样一般化. 面积法已经被推广到其它几何领域的定理证明中, 如立体几何、 Minkowskian 几何、Bolyai-Lobachevsky 几何、Riemannian 几何.

基于类似的思想, 一个新的几何量 —— 全角被引入到几何定理证明中来 [30]. 全角法虽然没有面积法强大, 但对某些复杂的几何定理能够生成比面积法更简短漂亮的证明.

Richter-Gebert J. 在 [112] 中提出了一个用于证明射影几何定理的基于括号代数的算法. 该方法的主要思想是把定理的假设部分和结论部分表示为关于括号的代数关系, 从而把定理的证明转化为对括号的代数计算. 射影几何中很多困难的定理通过此方法都得以证明, 而且证明过程也很简短.

AI 方法

除了上述基于代数计算的方法外, 还有另一类几何定理机械化证明方法 —— AI 方法. 这类方法的主要思想来自于公理化方法, 将几何公理与待证明的几何定理用逻辑语言形式化地表示, 然后依据逻辑推理规则, 通过搜索公理集与重写技术来构建证明树使得所有的叶节点都是公理集中的公式. 这种方法事实上是定理证明的通用方法, 并非局限于几何定理证明.

在实际应用当中, 人们研究更直观的, 更有效率的方式. 这方面最早的工作见于 1960 年 Gelernter H. 等的文章以及他们设计的几何机器, 所使用的方法被称为后推链法. 将待证明的几何定理的结论作为目标, 通过搜索建立好的几何公理集找到能够推导出此目标的假设, 将这些假设作为证明的子目标重复前面的过程, 直到最后的子目标都恰好是待证明定理给出的假设. 后来人们又提出前推链法, 即从假设出发搜索可能推出的结论. Nevins A. J. 将前推链和后推链结合 (重点是前推链), 虽然对证明方法作了许多改进, 但仍只能证明一些简单的定理. 近年来, Balbiani L. 等学者进行了大量研究, 将项重写等逻辑演绎推理技术应用到几何定理机械化证明上来, 并证明了一些相对简单的几何定理.

Fèvre 把经典的一阶逻辑和代数计算方法相结合建立了混合推理系统 [42] 并可以生成容易理解的证明.

周等学者将演绎数据库技术应用到几何定理证明中并取得了很大成功, 不仅证明了很多如 Miquel 定理 (该定理难于用代数方法证明) 的复杂几何定理, 而且还能自动发现新定理 [31]. 该方法可以发现几何构型中的不动点, 即能根据给定的一组规则发现由这些规则推导出来的所有几何性质, 从而提高了算法终止的效率. 同时通过将演绎数据库结构化大大缩小了其规模, 从而使搜索更有效率.

1.2.2 相关软件与项目

近 35 年来, 许多动态 (交互式) 几何软件 (Dynamic Geometry Software) 得到了广泛的研究与开发. 依据软件所提供的作图指令[注1], 用户可以使用鼠标在屏幕上绘制精确的满足特定约束关系 (如平行、垂直等) 的几何图形, 所有的构造步骤都被记录下来并可以随时重新构造. 所得到的图形不是静止的图片, 而是动态的, 即当用鼠标拖动图形中的某些对象时, 整个图形将实时地快速地重新绘制并依然保持其中的几何约束关系. 正是由于这种图形的动态性和精确性, 我们可以直观地观察当它变化时, 其中的某些对象是否仍然满足某种关系, 从而激发人们的思维得到相关的猜想. 大部分动态几何软件所提供的作图方式都是构造式的, 即几何图形的生成需要按照次序一步一步地执行相应的作图指令, 每一步构造都要依赖于上一步的结果; 有一些动态几何软件则可以通过对几何构型所蕴含的约束关系求解自动生成相应的动态几何图形, 如 GEOTHER [127]、Geometry Expressions [50]、JGEX [67]. 除了欧几里德平面几何图形外, 有些动态几何软件还可以构造立体几何图形 (如 Cabri 3D [18]), 非欧几何图形 (如 Cinderella [32]), 并提供脚本编程与宏扩展功能, 构造分形几何图形 (如 Cinderella、GeoGebra [46]、GCLC [45]). 另外, 随着网络的普及, 许多软件都提供了 Web 版本, 基于 Web 脚本的跨浏览器平台的动态几何软件包也应运而生, 如 JSXGraph [70].

除了生动地显示动态的几何图形之外, 动态几何软件一般还具有几何计算与推理的功能, 例如 Geometry Expressions 可以根据约束关系对几何量进行符号计算; GeoGebra 可以进行几何量的度量; Cinderella 可以进行非欧几何计算并实现了括号代数的证明方法; GCLC 实现了面积法; Geometry Explorer [49] 实现了全角法并将证明过程通过图解的方式来显示; GeoProof [51] 实现了面积法并通过使用 Coq 系统交互式地构造几何

[注1] 本文中的作图指令是指动态几何软件所使用的图形绘制命令, 作图指令的执行称为构造.

定理的形式证明; GEX (Geometry Expert) [48] 及其 Java 版本 JGEX [67] 实现了前文所述的绝大部分几何定理机械化证明方法, 包括吴方法、Gröbner 基方法、面积法、全角法、演绎数据库法等并在图形上动态地显示证明过程; 张景中院士等开发的超级画板 [144] 实现了几何的符号计算、数值计算以及演绎数据库法等诸多推理方法, 可以有效地证明及探索几何定理.

基于图形的动态显示和强大的计算功能, 很多软件还提供其它丰富的功能, 如构造几何变换, 生成轨迹, 动态显示函数图像以及图形的动画效果, 物理模拟, 生成高分辨率的图片等. 动态几何软件使得图形变得更加生动, 并且不局限于几何构型的呈现, 因此在许多学科的教育和研究中都得到了广泛的应用. 超级画板是目前功能最全面和强大的动态几何软件, 它用智能的作图方式代替了一般动态几何软件中菜单式的作图方式, 使得几何作图更为简洁, 而且提供了强大的图形显示与计算功能, 交互式推理功能, 程序和文档编写环境等为制作课件提供了有效的工具. 超级画板成为几何、数学、物理、概率等学科教学的重要辅助工具, 有着非常广泛的应用. 对于每种软件所实现的功能比较, 感兴趣的读者可参阅 [87].

目前动态几何软件已经相当丰富, 所提供的功能也相当完善. 一些项目则关注如何将几何定理或 (证明) 问题标准化, 并建立可供不同软件共享与重用的几何资源库.

GeoCode [54] 是一个通用的用于证明几何定理的表述语言, GeoProver [52] 可以将使用这种语言表述的定理翻译成不同计算机代数系统 (Maple、Mathematica、MuPAD、 Reduce) 所表示的代数形式, 进而使用坐标代数方法进行机械化证明. 该项目收集了 250 个使用 GeoCode 表述的几何定理.

GEOTHER [127] 是一个基于 Maple 开发的进行机械化证明和处理几何定理的软件包. 它提供了一种标准的谓词语言来对几何定理进行表述, 同一个表述可以被自动翻译成自然语言 (汉语或英语) 表述, 一阶逻辑语言公式或者用 Maple 语言表示的代数表达形式, 并且可以自动生成动态的几何图形, 使用多种坐标代数方法进行机械化证明, 并将结果输出到文档以便打印. 该软件包同时建立了一个列表包含多个定理的表述.

GeoThms [109] 是一个整合了动态几何软件 (GCLC、Eukleides)、几何定理证明器 (GCLCprover) 及几何问题库的网络框架. 用户可以容易地浏览问题列表 (包括问题的陈述、例子、证明等), 并在统一的框架下使用定理证明器, 动态几何软件来探索几何猜想, 并把它们添加到问题列表中.

Gool [83] 是一种符号化几何计算与推理的面向几何对象语言. 它独立于图形而使用确定的数据或不确定的符号来构造平面或其他几何空间中的对象, 并且可以实时地修改符号所表示的几何对象的参数值等, 陈述它们之间的相互关系. 基于这种语言的系统可以对不同的情形进行区分, 同时采用逻辑演算, 代数化简等手段来进行严格精确的几何计算和推理, 证明和发现几何定理, 检测几何约束的相容性等.

INTERGEO [65,121] 是一个为期 3 年的欧洲项目, 目的在于建立一个标准的, 跨不同动态几何软件的, 跨不同语言的, 跨文化的几何教学资源共享平台. 它提供了一个描述动态几何图形的标准格式 [1,40] 并且由此实现了不同动态几何软件之间的交互, 通过对所创作的动态几何教学课件进行课程信息标注并开发相应的几何本体 GeoSkills 和搜索引擎实现了资源的共享 [85], 同时为软件提供者、资源创建者、教师与学生建立了交流讨论反馈的环境以增强平台运行的实用性.

TGTP (Thousands of Geometric problems for geometric Theorem Provers) [118] 是一个基于 Web 的为测试和评估不同几何定理证明器而建造的标准几何问题库系统. 该系统将问题的表述自动转化为被测试证明器的内部表示, 并且对每一个问题搜集了使用不同证明方法所耗费的时间, 成功与否等数据, 通过对这些数据进行统计计算来比较不同证明器的优劣和特点. 目前该问题库已经包含了 170 个几何定理并对 Coq (面积法)、GCLCprover (面积法、吴方法、Gröbner 基方法) 做了测试与分析.

这些项目虽然涉及到数学知识管理所关注的一些问题 (如几何定理、问题、构型的标准化), 但其出发点都以解决几何问题为主要目的 (如同一表述可以使用不同的几何定理证明器进行机械化证明, 可以使用不同的动态几何软件来呈现和处理等), 并没有从几何知识的角度出发来研究几何表述[注1] 的标准化. 另外, 知识管理所关注的许多其它方面也没有涉及, 如几何知识的结构化、查询、浏览等.

1.3 本文研究的动机、结果及意义

1.3.1 主要内容

几何学作为一个古老的数学分支已经发展了两千多年, 人们也积累了大量丰富的几何知识. 然而到目前为止, 人们关注更多的是怎样应用计算机来解决几何学中的问题, 如几何定理的机械化证明, 几何构型的动态呈现. 事实上, 人们更擅长的是创造性推理, 而

[注1] 本文中的几何表述是指使用几何语言从几何意义上对表达对象的陈述.

计算机则长于管理. 数学知识管理领域正在活跃的发展当中, 但研究的对象集中于一般化的数学知识, 并没有关注到专门的几何领域. 几何学研究的是从现实世界抽象出来的有“形”对象, 并且可以引入抽象的“量”来描述这些对象之间的关系. 这就使得几何知识拥有区别于其它数学知识的特殊性: 既包含了具体的“形”又包含了抽象的“量”. 相比于其它数学领域, 计算机更可以同时发挥其在图形显示与计算方面的强大功能. 另一方面, 几何学机械化的研究方兴未艾, 有效地利用已有的方法和软件工具也将为几何知识的管理提供有力的条件和帮助. 因此, 我们需要开辟新的思路, 研究新的方法来管理几何知识.

数学知识管理的一个主要目标是要应用相关的理论方法开发完善的管理工具以满足人们对于数学知识共享和使用等方面的需求. 因此, 需要研究人们对于数学知识都有哪些需求, 要提供怎样的方式对数学知识进行管理以满足这些需求. 尤其对于几何知识, 需要探索如何合理地构建及有效地利用几何知识资源及其结构, 通过何种方式来整合几何知识与现有的几何软件系统或工具.

完成这样一个目标并不容易. 首先, 由于几何学的研究同时包含“形”的推理与“量”的计算, 几何知识也通常包含多个方面, 具有不同形式, 因此需要全面地整理与分析几何知识在不同环境下的表述方式, 以便为实际应用提供有力的数据支持; 其次, 几何知识需要以合理的方式进行组织与结构化, 以便为检索、浏览、文档构建等应用提供支持. 因此需要考察它们之间所存在的各种关系, 并据此进行归类整理、组织规划, 使得几何知识按照一定的结构发展扩充, 从而可以有效地对其进行控制与操作.

几何知识需要被形式化以便于计算机处理. 几何知识的表述基于大量的几何概念, 而几何概念通常依赖于具体的图形来建立其明确的含义, 因此需要对几何概念做深入的分析, 对不同情形作出明确的区分以期摆脱对图形的依赖, 从而达到形式化的目的; 另一方面, 几何知识涉及到量之间的代数运算, 因此将几何知识形式化还需要研究如何将几何元素与代数元素在统一的框架下表述, 从而进行有效地处理; 几何知识通常都是针对几何构型的, 而几何构型可以通过两种方式进行表述: 一种是*构造式*, 即一般的动态几何软件中所使用的构造方式; 另一种是*约束式*, 即不分次序地陈述其中所包含的几何约束关系. 表述的方式将直接影响到其对应的几何图形的动态显示以及几何问题的解决方法, 因此将几何知识形式化也需要研究如何将构造式与约束式的几何构型在统一的框架下表述, 并且形式化的表述应当尽可能与日常使用的表达方式接近以便于人们的阅读、使用与理解.

通常几何知识的表述都很简洁, 这是由于它应用了大量相对复杂的几何概念. 而在几何问题的解决过程中, 需要将其所应用的几何概念用其它同义的 (或等价的) 表述来代替从而将问题转化为易于处理的形式, 这种问题转化的思想在日常的数学研究中也经常使用. 事实上, 几何机械化推理的大部分方法也是基于这种思想的, 例如坐标代数方法是将几何问题转化为关于坐标变元的代数消去问题, 非坐标代数方法是将几何问题转化为关于几何量变元的计算问题. 几何学的机械化方法可以辅助人们进行几何推理, 解决几何问题. 如 [54] 所指出的, 尽管人们通常使用提法“几何问题的自动解决”, 但从人机交互的整个过程来看, “机械化方法”并不是“自动化推理” (吴在 [138] 中也强调了这一点) , 因为人们在传统几何文献中所见到的可读的几何问题陈述都需要转化成另一种计算机可以理解并操作的形式, 然后才能使用前述的机械化方法进行处理. 例如, 在 [26] 一书中所有给出的通过坐标代数方法证明的几何定理都包含定理的自然语言 (英语) 表述, 由作者将定理转化而成的象 LISP 语言表述的形式以及由此通过计算机自动转化而成的相应的代数形式. 前一种转化包含了几何概念的替换以及合适坐标的选取, 目前这只能由具备相关几何知识的有经验的人来完成, 以使得转化后的代数问题易于计算机求解. 因此, 我们需要研究这种转化的过程并将其算法化, 从而使得几何推理真正达到自动化, 也使得现有的几何软件可以直接处理文献中的“可读”几何问题. 所谓的可读是指表述需要接近自然语言以便于人们理解交流, 同时又是形式的以便于计算机处理. 这种自动转化依赖于几何知识的形式化表述.

我们目前以欧几里德平面几何知识作为考察的对象, 对以上问题进行了认真细致的研究, 现将工作的创新点和结果陈述如下:

- 参考并借鉴一般数学知识管理的思想和方法, 针对几何知识的特殊性, 提出了一个新的研究方向 —— 几何知识管理, 阐述了其研究的主要内容和需要解决的问题;
- 提出了一套管理几何知识的方法和技术, 为此方向的研究制定了初步的理论框架;
- 应用所提出的方法和技术, 提出了采用动态教科书的形式来表示和管理几何知识的新思路, 设计完成了一个几何知识管理系统 —— 电子几何教科书系统, 并阐述了其设计方法和原理;
- 设计完成了一个服务于几何自动推理与计算、自动作图、自动翻译、同义转化、不同版本文档生成的几何知识库;
- 根据几何知识的特点与应用的需求, 研究并设计了用于表述几何概念、定义、

命题、问题等几何知识的形式语言 —— 几何描述语言 (Geometry Description Language 或 GDL);

- 研究并实现了 GDL 表述[注1] 的自动同义转化, 进而开发了几何描述语言与几何定理证明器 GEOTHER 和动态几何软件 GeoGebra 的交互接口, 从而实现了几何定理 (GDL 表述) 的自动证明与相应动态几何图形的自动绘制;
- 实现并构建了一个包含近千条几何定理与定义, 并具有创建、修改、删除、浏览以及检索知识数据功能的几何知识库;
- 实现了电子几何教科书系统, 其功能如下:
  - 通过对话框创建教科书所包含的知识内容及其相互关系;
  - 交互式地构建树形结构的教科书, 并可以增添、插入、删除、修改、重构教科书的内容;
  - 以合适的粒度存储、维护、重用教科书的内容和结构;
  - 在教科书构建的过程中, 可以实时地检测其叙述结构的一致性 (内容安排的顺序是否合理)、内容的完备性 (是否全面) 和冗余性 (是否重复);
  - 自动发现所创建知识内容之间的特定关系;
  - 动态地呈现知识内容之间的相互关系;
  - 实时地改变系统的工作语言 (英文或中文), 并可按需要自动生成教科书局部或者全部的样式化的英文或中文文档;
  - 自动生成文档中关于概念、图形、章节等之间的交叉引用;
  - 所构建教科书中的命题可以通过调用 GEOTHER 进行自动证明, 并且可以通过调用 GeoGebra 自动绘制其对应的动态几何图形.

教科书一直以来都是系统化领域知识的载体, 按照一定的逻辑顺序对内容进行组织使得读者可以循序渐进地摄取知识. 我们采取动态教科书的形式来表示和管理几何知识是一个新的尝试, 它改变了传统教科书仅静态地呈现内容的文档模式, 通过软件系统的模式对教科书的内容进行织和管理, 辅助人们直观方便地浏览教科书的结构并实时地改进其内容. 知识管理功能使得这样一本教课书的创建、维护和使用更有效率: 一方面依据教科书内容之间的相互关系可以自动检测它们安排的顺序是否合理, 是否全面或者存在重复, 从而有效地辅助教科书合理地构建和维护; 另一方面, 教科书的内容可以表示成

[注1] GDL 表述是指应用几何描述语言得到的表述.

多个版本并对其进行细粒度地存储和组织, 不同的版本适用于不同的应用. 人们可以自由地贡献、共享以及重用多版本内容, 从而提高教科书的创建、修改、更新和改进效率. 得到的教科书不仅可以打印成传统的文档, 其内容还可以被计算机自动处理与操作: 其中的几何命题可以被自动证明, 从而辅助验证教科书内容的正确性; 其中的几何构型可以动态地呈现, 使几何知识更形象直观便于理解, 这也使得原本“单调”的教科书变得生动. 另外教科书内容之间的相互关系可以通过图形的方式呈现, 辅助人们理解和把握知识的结构.

虽然我们目前考查的对象限于欧几里德平面几何的范围, 但所研究的问题、内容及提出的解决方案却是对于构建任何一本类似的教科书都适用的.

1.3.2 结构安排

本文所述工作的基本思想来自于王东明教授, 作者对这些思想进行细化分析阐述并完成了电子几何教科书系统的设计与实施.

第二章提出了几何知识管理这一新的研究方向, 从几何学区别于其它数学学科的角度阐述了研究的动机、包含的主要内容和问题, 并分析了几何知识管理与一般的数学知识管理之间的联系和区别. 从构建几何知识库和设计知识库与外界接口两方面探讨了管理几何知识的方法和技术, 是后续章节的思想基础和指导. 这一章可看作是全文内容的概览.

第三章应用第二章所提出的几何知识数据的结构化以及维护方法, 通过获取整理几何知识数据并分析构建知识对象以及知识图, 具体地实践了几何知识库的创建过程, 从而设计完成了几何知识库的数据存储结构. 最后展示了几何知识库的基本功能, 并说明了实现中所使用的技术、工具和方法.

第四章提出了电子几何教科书系统的设计动机、研究的内容和意义, 展示了系统的架构, 详细地分析和研究了几何知识的表述, 尤其是设计了一种几何描述语言, 以此为基础解决了几何表述的自动同义转化问题和几何知识对象之间关系的自动发现问题并给出相应的算法. 另外, 通过利用几何知识库中的元知识数据, 研究并解决了教科书的结构一致性、完备性以及冗余性的自动检测问题.

第五章主要展示了电子几何教科书系统的实现方法与结果, 包括所构建的教科书知识库、系统的人机交互界面、几何教科书及其元知识数据的动态呈现、几何描述语言的计算机表示方法以及应用几何表述的自动同义转化方法来实现教科书中命题的自动

证明与其动态几何图形的自动绘制.

第六章是全文的总结, 并针对其中未完善的部分以及待解决的问题对未来的工作做一个展望, 包括几何知识表述的形式语言的完善, 对几何知识进行自动处理的相关问题以及将电子几何教科书系统应用于教育而需要发展的功能.

本文第二章的内容是基于作者与王东明教授合作的结果 [24,25], 第三章的部分内容是基于作者与黄荧女士及王东明教授合作的结果 [23], 第四章与第五章的大部分内容是作者的结果 [21,22].

第 二 章　几何知识管理

2.1　引言

目前数学知识管理的研究主要集中在算术、代数和微积分等数学领域. 在这些领域中, 数学理论可以通过使用公理化方法和现代逻辑理论机械化地或者交互式地构造与组织, 这也是通常数学知识管理研究所使用的方法. 尽管 Euclid 和 Hilbert 的公理化方法极大地促进了几何学的发展, 但是针对几何知识的计算机辅助管理研究还几乎没有开展, 主要原因是使用通常的公理化方法机械地证明几何定理或者交互式地构造形式证明的效率不高. 一方面, 正如吴在 [138] 中所指出的, Hilbert 的公理化体系远远不够严格, 要建立一个与 Hilbert 公理化方法类似的, 又能严格准确地描述几何的公理系统是相当困难甚至是不可能的. 我们通常见到的 (书本中的) 几何公理、定理等描述都假定所考虑的几何体和几何关系是处于正常的一般化情形, 而不是带有特殊性的退化情形. 例如, 说两条直线相交隐含地假设这两条直线不重合为一条直线, 说一个三角形通常会隐含地假设它的三个顶点处于不共线的位置 (即非退化条件), 否则三角形就退化为一条直线, 这样的三角形的内心、外接圆等对象就变得无意义. 换言之, 通常的几何定理都是在某些非退化条件的约束下才成立的. 当考虑如何在计算机上表示几何知识, 进行几何计算和推理等操作时, 不可避免地要去面对那些特殊的情形. 由于这些条件通常都是隐含的, 计算机无法预知并处理, 因此几何知识的精确性无法得到保证. 另一方面, 通常的 (书本中的) 公理体系不适用于形式化几何知识, 因为其中几何定理的证明一般需要图形的辅助, 例如, 在证明平行四边形 $ABCD$ 的对角线互相平分时, 会用到平行线内错角相等这个性质, 然而这个性质的使用前提 (A 与 B 在对角线 CD 的两侧) 是通过对图形观察而得到的. 虽然一些几何理论, 例如 Tarski 几何和射影平面几何已经在证明助手 Coq 中被形式化地构造而且其正确性也得到了保证 [89,98], 但这种形式证明的构造并不是平凡简单的而通常需要使用合适的技巧和策略才能完成, 因此使用这种交互式的证明方法来管理几何知识的适用范围比较小, 效率比较低. 因此, 一般的数学知识形式化方法并不适用于管理通常的几何知识, 而应当研究探索新的思路和手段.

几何学建立于从客观世界抽象出来的图形对象基础之上, 几何计算、推理、可视化

等需要逻辑演绎、代数计算[注1]、图形绘制等工具. 相比于其它数学学科, 只有几何学涵盖了如此多有趣迷人的定理, 并且随着人们对几何计算推理方法的深入研究, 这些定理都可以被机械地证明, 同时得到定理成立的非退化条件 [138] (事实上, 自从吴方法在几何定理机械化证明领域得到成功应用以后 [137], 几何学自动推理成为自动推理界比较成功和活跃的领域), 几何构型对应的动态几何图形可以在计算机上交互式地精确构造甚至自动生成. 因此, 几何学是内容极其丰富的数学学科, 并且解决几何问题的软件工具也如此丰富与强大, 所有这些都为研究几何知识的管理方法提供了条件和支持. 在此过程中, 计算机在数值与符号计算、图形处理、数据处理等方面的能力可以得到充分地使用和发挥, 从而为考察数学知识管理的新观念和新方法提供一个很好的平台.

几何学是研究图形的大小、形状、相对位置及空间性质的数学分支. 在几何学的研究过程中, “量”与“形”是紧密相连的, 尤其是在解决复杂的几何计算问题 (如解析几何中基于坐标量的计算), 应用代数方法进行几何定理机械化证明 (如坐标代数法中基于坐标量的计算, 非坐标代数法中基于几何不变量的计算), 以及在计算机上精确地显示几何图形 (位于屏幕的坐标系统之中) 时. 因此, 管理与使用几何知识应不仅仅局限于几何学本身, 还应该包括相关的代数元素 [128, 129]. 几何知识应该涵盖几何形式 (刻画几何意义), 相应的代数形式 (刻画代数意义) 以及图形呈现 (刻画现实世界) 三方面的内容. 例如, 要全面地刻画与圆有关的几何知识, 需要陈述语句“以点 O 为圆心 r 为半径的圆”或者“到点 O 距离为 r 的点的轨迹”, 代数方程 $(x-a)^2+(y-b)^2=r^2$ (其中 (a,b) 为点 O 的坐标), 参数方程 $\{x=\frac{1-t^2}{1+t^2}, y=\frac{2t}{1+t^2}\}$ (其中 t 为参数), 以及在计算机屏幕或打印机中呈现圆的图形 (或轨迹) 所需要的指令表述. 按照这样的观点, 几何知识可以分为如下类型.

- 陈述型知识: 对几何事实或事物等的陈述, 包括以下不同形式:
  - 几何形式: 使用几何语言从几何意义上对几何知识的陈述, 例如存在于几何文献当中的概念定义、几何构型、几何命题、证明过程、简介或注释等;
  - 代数形式: 使用代数语言从代数意义上对几何知识的陈述, 例如几何命题在引入的坐标系统中所对应的代数形式 (即代数化), 或者关于某些几何不变量的代数形式;
  - 作图形式: 使用几何作图语言从图形绘制的方法上对几何构型的陈述, 例如应用直尺和圆规绘制几何图形的步骤描述;

[注1] 据我们所考察, 应用计算机来研究几何学的过程中, 代数计算比逻辑演绎使用得更广泛.

- 过程化知识: 用于描述做事的方法流程, 并可以在几何软件中实现为程序, 包括:
  - 代数化知识: 将几何概念转化为同义的 (或等价的) 代数形式的规则, 例如由一个圆得到其二次方程的规则;
  - 图形化知识: 将几何概念转化为其所对应几何图形绘制方法的规则, 例如由一个圆得到在动态几何软件中绘制这个圆的作图指令;
  - 问题解决知识: 解决几何问题的算法、方法等, 例如圆面积的计算公式, 证明几何定理的机械化方法.

正如在 [19] 中所讨论的, 数学知识管理的涵义依赖于看待数学的角度以及使用数学知识的目的. 基于前述几何知识区别于其它数学知识的这种多样性, 参考数学知识管理的主要思想, 我们提出一个专门化的研究方向 —— 几何知识管理, 研究如何利用先进的计算机科学与技术，提出新的思路和方法，设计开发整合的*几何知识管理系统*来维护、处理、呈现不同形式多种版本的几何知识，并提供代数与几何计算、图形构造与绘制、自动推理与知识发现等功能，以支持人们方便高效地创造、共享、访问、使用几何知识. 其研究内容主要涉及到知识库与接口两方面 (见图 2.1).

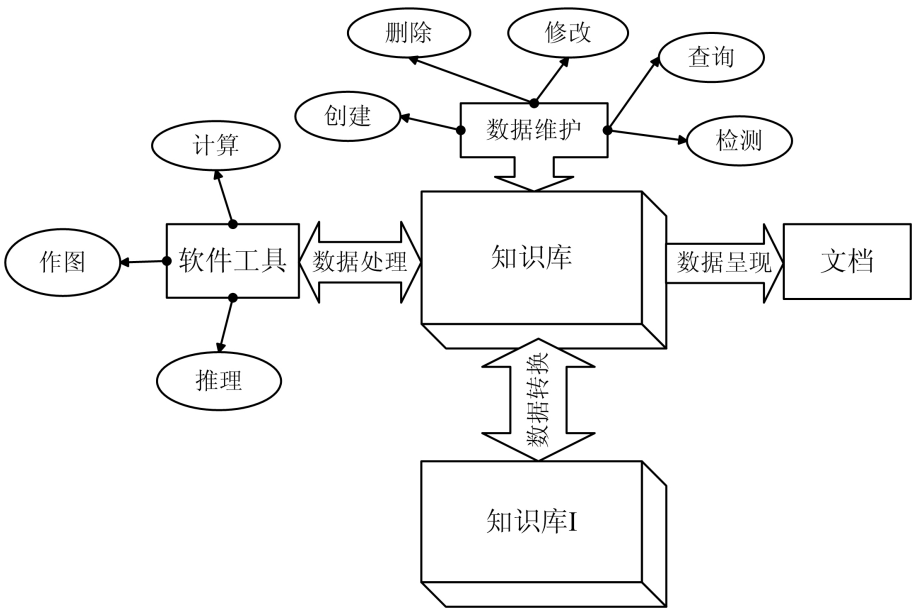

图 2.1 **几何知识管理系统架构**

知识库是一种用于管理某领域知识的特殊的数据库, 是几何知识管理系统的核心. 几何知识库的设计与建造涉及到如下三方面主要内容:

- 获取与整理: 如前所述, 不论表现形式还是具体内容, 几何知识都具有多样化的特点. 要研究几何知识的管理方法, 首先需要对几何知识进行全面地获取分析及整理使其表述规范化, 以保证其可靠性与无歧义性;
- 语言与表示: 知识内容需要使用某种语言表述以形成数据. 一方面, 它们需要被计算机处理以支持自动化功能 (如语句转化、定理证明、图形绘制), 因而需要被形式化, 即用某种具有精确语法规则的形式语言来表述. 形式语言的设计应定位于表述的逻辑层面, 即独立于具体的实现; 基于这种语言, 它们需要被表示成某种具体的格式才可以被软件工具识别, 表示的格式应定位于表述的实现层面, 即便于程序的使用操作. 另一方面, 它们还需要被表述成非形式的自然语言并使用可样式化的计算机可读格式来表示, 以支持人们浏览与阅读 (如同传统的文档);
- 组织与结构: 知识内容之间不是独立的, 而是相互联系地处于相应的上下文语境 (如显示的章节结构以及隐含的逻辑关系) 之中, 因此需要被结构化, 以便为管理知识的过程提供元信息. 为了合理地对几何知识数据进行组织和规划, 我们需要对元知识 (即系统化结构化几何知识的知识) 进行建模. 元知识指明了多版本知识内容的使用范围、层次结构、相互关系以及知识积累发展的轨迹. 建模元知识最主要的任务是建模知识的个体, 它们的属性以及它们之间的关系. 几何知识的结构化可以看作是关于几何知识的元知识建模.

知识库不仅应存储几何知识数据, 而且需要存储关于几何知识的元知识数据, 以便为管理过程提供数据支持. 我们将几何知识数据连同其元知识数据统称为*知识数据*. 知识数据与其使用者 (人和机器) 之间的交互和联系需要基于知识库的接口. 这些接口定义了管理过程中对知识数据的应用需求, 主要包括:

- 数据维护: 知识数据需要被操作 (创建、修改、删除、检索) 以便可以实时地扩充和改进知识库, 并且需要对知识库进行正确性、冗余性、完备性检测以保证其合理与可靠;
- 数据呈现: 知识数据需要通过显示工具 (如浏览器、打印机、信息可视化工具以及动态几何软件等) 以传统可读的样式呈现 (经过必要的转化) 出来以便于人们阅读或探索;
- 数据转换: 知识数据需要从一种表示格式转化成另一种表示格式以便不同的知识管理或者知识处理系统重用;

- 数据处理: 形式化的几何知识数据需要被转化并处理, 例如, 使用特定的数据进行几何计算, 自动或者交互地证明几何定理, 探索以及呈现特定的数据, 将其转化为自然语言语句等.

数学知识管理系统通常管理形式化的 (或者半形式化的) 应用逻辑语言表述的数学知识. 然而大部分有效的几何推理方法和工具都是基于代数表达式的符号计算而并不操作几何知识 (如定理、引理、定理等) 表述本身. 我们从以下三个方面分析一般的数学知识管理与几何知识管理之间的联系和区别.

- 对于表示、呈现以及转换几何知识, 数学知识管理的相关技术和标准可以被应用, 如 MathML、OpenMath、XSLT、WYSIWYG 创建工具, 动态几何软件中使用的作图指令, 几何定理证明器中使用的谓词描述, OpenMath 格式中的平面几何词汇以及 INTERGEO 项目所制定的动态几何图形的标准格式都是已存在的几何构型的表示格式. 然而在知识的层面上 (尤其是关于几何构型的定义、定理、转化规则, 类似于 OMDoc 格式在对象层次之上的陈述层次) 来进行几何表述的方法和技术是目前所缺失的. 为了整合已有的软件工具有效地处理几何知识, 满足于这个层次需求的统一的形式化语言需要被设计和开发.
- 对于结构化几何知识, 一些数学知识管理项目所提出的方法可以被借鉴, 如 OMDoc、MathDox、MOWGLI、INTERGEO. 由于几何知识具有多种形式并且包括多种版本, 怎样组织几何表述潜在的组成内容 (包括代数形式、作图指令、非退化条件等), 尤其是处理这些组成内容之间的关联、基数、依赖关系以及如何建模这些表述的结构以满足实际应用的需求 (如文档或课程的生成与验证) 都是需要被深入考察的. 因此, 我们需要开发专门的技术来控制几何知识内容的结构复杂性以及在管理过程中维护这些组成内容.
- 不同于一般数学知识管理中由领域知识库支持的推理过程, 几何推理与知识探索需要独立于几何知识库的专门工具来进行代数操作. 几何图形也可以通过应用已有的动态几何软件来构造和呈现. 因此, 为了进行几何计算、推理与可视化, 不能通过应用推理规则来处理和生成知识表述的方式, 而需要开发专门的方法将几何知识库中的几何表述转化为可以被已有的几何软件工具 (和计算机代数系统) 执行的语句.

针对以上几何知识管理的研究内容, 通过总结分析利用现有的资源和工具, 我们提出下述管理几何知识的方法和技术.

2.2　构建几何知识数据

2.2.1　获取几何知识

陈述型几何知识的主要来源是几何学文献、数字图书馆和电子数据库等 (如 Wolfram MathWorld [134]). 有些几何知识隐含在几何问题的解决方法之中而并没有被明确地提出, 如有关几何问题转化方面的知识, 几何定理的非退化条件等, 这些需要从有经验的几何专家那里获得. 随着计算机技术的发展, 又出现了许多新的几何知识并已经被应用到几何软件的设计开发中, 如用于解决几何推理和计算问题所提出的各种机械化算法或者方法, 动态几何软件所使用的图形绘制方法等. 另外由于整个数学的发展, 各学科的交叉和相互影响也在不断地加强, 因此几何知识的规模也将会快速地扩大, 形式也将会不断地发展变化. 几何知识的获取以及管理最终是要为人们服务, 因此应该以需求为导向, 有针对性有目的地获取, 避免造成资源的浪费. 另一方面, 现有的几何机械化方法可以用于发现新的几何定理, 尽管能力有限, 但相应的软件工具也可以作为获取几何知识的一个途径.

2.2.2　规范几何知识

由于人们使用习惯以及文化等的差异, 所获取的几何知识会在表达以及概念符号的使用上存在着不同 (或者不精确). 一方面, 不同的符号可能表达相同的含义. 例如, 线段 AB 的长度可以表示为 $|AB|$, 也可以表示为 $\overline{AB}$. 因此, 建立开放协作式的符号字典 [84] 会有助于知识的自适应呈现以及有效地交流; 另一方面, 相同的符号或概念在不同的情形下表达不同的含义. 例如, 通常所使用的半径的概念是不精确的, 因为半径既可以表达一条线段也可以表达这条线段的长度; $\angle$ 所表达的含义也是模糊的, 因为它既可以表达一个角 ($\angle ABC$ 的扇面), 又可以表达这个角的大小 (如 $\angle ABC = 60°$). 另外, 相同的几何知识可以通过不同的语句表述. 例如, 帕斯卡定理可以叙述为“令 A、B、C、D、E、F 为共圆的六个点, 则 $P = AB \cap DE$、$Q = BC \cap EF$、$S = CD \cap AF$ 在同一条直线上”; 也可以叙述为“若一个六边形内接于圆, 则对边交点共线”. 因此, 为了使所获取的几何知识可以被准确地交流, 形式地访问以及可靠地转化, 应当建立统一的一致的规范来约束

和整理几何知识.

2.2.3 表述几何知识

经过规范化的几何知识需要使用某种语言来表述以形成数据. 所谓语言, 即用来表述几何知识的有限长符号串, 其中的符号需要取自预先定义好的符号集, 符号的排列需要遵从一定的语法规则. 例如, 自然语言的表述需要符合自然语言的语法; 几何构型的表述可以遵从不同动态几何软件中作图指令的语法. 不同语言的表达能力以及可进行的处理和操作都不相同. 通常我们所获取的规范的几何知识都是使用自然语言表述的, 可以很好地被人理解、交流和整理, 但并不能有效地被计算机理解、处理和使用. 因此需要设计具有精确语法规则的形式语言来表述几何知识. 这种形式语言的设计依赖于应用的需求而没用统一的方法, 基本原则是简洁和自然, 方便人们阅读和使用. 我们将在第 4.3 节来详细讨论几何知识表述语言的设计并展示一种用于几何表述的形式语言 (几何描述语言) .

另一方面, 同一种语言在计算机上可以使用不同格式来表示 (例如可以仅是语言本身的文本字符串, 也可以是用标注语言表示的文档). 不同的格式会影响数据处理的方法和效率, 因此需要选择一种合适的格式来表示、存储以及处理几何知识数据. 我们将在第 5.4.1 节来详细讨论几何描述语言的计算机表示.

2.3 构建元知识数据

这一节, 我们讨论领域知识[注 1]的元知识的建模方法.

2.3.1 封装知识对象

领域知识数据尽管是分散的, 但相互之间有着内在的联系. 为了控制这些联系的复杂性而便于管理, 一个经典的策略就是根据应用的需求采取合适的粒度将密切关联的领域知识数据封装起来作为管理过程的操作对象. 如果粒度太小, 那么管理过程将会变得很复杂而不可控制; 相反, 如果粒度过大, 那么管理过程将会趋于平凡而变得无意义. 一些标注语言 (如 OMDoc) 按照多种层次粒度将数学文档的文本分类划分为不同的个体单位. 我们将这种可以被识别、区分、理解以及处理的领域知识单位称为*知识对象*. 例如, 三角形内心的定义是一个知识对象, 它解释了内心这个概念的含义; 毕达哥拉斯定理 (见

[注 1] 这里专指几何知识.

附录 **A.)** 是一个知识对象, 它描述了关于一个平面几何构型的事实; 其证明也是一个知识对象, 它说明了该定理为何成立.

知识对象可以划分为不同类型, 使得管理过程的相关操作 (如创建、处理、交互) 可以统一地进行. 我们称这些类型为*知识类*. 例如, 三角形内心的定义属于*定义*这个知识类; 毕达哥拉斯定理属于*定理*这个知识类. 类似于 OMDoc 使用 Schema 来定义标注元素的内部数据结构, 每一个知识类都包含多个特定的*属性*用于指明知识对象使用的特征、约束以及版本. 每个属性都由*属性名*和*属性值*构成, 这些不同的属性名指明了知识对象可能进行的不同应用, 而属性值则表示知识对象在相应的应用中所使用的领域知识数据. 例如, 定义这个知识类包含如下属性: `knowledgeName` (用于显示定义对象的名字), `formalDefinition` (形式表述, 用于几何表述的转化与处理等自动化应用), `naturalRepresentation` (自然表述, 用于人们浏览查看此定义对象的含义). 内心的定义这个知识对象相应地拥有属性值"内心", "Definition(incenter(triangle(A::Point, B::Point, C::Point)), [concurrentpoint(innerbisectors(triangle(A, B, C)))], null) [注1]", "三角形内心是其三条内角平分线的公共点".

根据应用目的的不同, 确定几何知识数据封装的粒度并定义知识类的内部数据结构 (即包含哪些属性) 是建模元知识的第一步, 也是管理几何知识的首要任务.

2.3.2　制定知识图

为了给知识管理的操作提供语义支持, 需要对领域知识进一步结构化. 对信息进行语义建模的技术已经得到巨大的发展, 如语义网技术 [37], 本体技术 [56] 等. 就其本质而言, 语义包括两部分: 信息实体的分类体系与它们之间的相互关系. 知识对象将有意义和密切关联的知识信息封装起来成为领域知识的基本单位, 因此, 领域知识的结构化可以通过知识对象的结构化来刻画. 我们采用以下两种方法. 知识图受到 RDF 数据模型 [111] 的启发, 可以被看作是刻画知识对象结构的带有多重关系的二维图表, 定义如下:

定义 2.3.1. $\mathscr{N}$ 上的一个*知识图*是一个有序对 $(\mathscr{N}, \mathscr{L})$, 其中

(1) $\mathscr{N}$ 是用于封装领域知识数据的对象的集合, 其元素叫做*节点* (*vertices* 或 *nodes*);

(2) $\mathbb{L}$ 是用来标识节点之间关系类型的名字的集合, 其元素称为*标签* (*labels*);

[注1] 关于几何表述的形式语言, 我们将在第 4.3.2 节中详细讨论.

(3) $\mathscr{L} = \{A \to_\gamma B \mid A \in \mathscr{N} \wedge B \in \mathscr{N} \wedge \gamma \in \mathbb{L}\}$, 其元素称为*链接* (*links*).

知识图本质上是图结构, 通过向图中增加关系类型信息来描述元知识. 例如, 知识对象集合上的一个知识图包含链接 $A \to_\gamma B$, 其中 A 表示角平分线的定义, B 表示三角形内心的定义, γ 表示关系类型 contextOf, $A \to_\gamma B$表示了角平分线的定义提供了三角形内心的定义所需要的语境. 知识图最主要的组成部分就是链接, 一个好的知识图在于其中的链接可以全面准确地反映知识对象之间存在的各种关系, 因此深入分析与全面获取关系类型是制定实用的有价值的知识图的关键. 事实上, 这些关系类型本身还可能包含着各自不同的属性特征, 它们之间也仍然存在着层次结构和相互关系, 这些关系类型的元信息为知识图本身提供了语义支持, 也是元知识的一部分. 并且从知识检索、维护等应用的角度看, 研究如何来组织和管理关系类型, 必将能更加有效地满足知识管理的需求. 因此, 制定知识图, 不仅要获取标注知识对象之间的链接, 而且要同时分析考察所使用的关系类型的结构 (通过构建 RDFS 和本体). 这样获得的知识对象集合上的知识图, 才能更精确地刻画以每个知识对象为中心, 与其它知识对象相联系的元知识.

知识对象集合上的知识图反映了元知识的一个侧面, 另一种用于组织知识对象的方式是使用类别. 类别用于阐明具有相同主题的成员之间的聚集关系, 在导航、浏览、呈现领域知识中扮演着重要的角色. 例如, 教科书中的章节等都属于类别. 一般来讲, 类别的结构类似于树的结构, 其叶子节点是知识对象, 其它节点指明了类别的主题, 定义如下:

定义 2.3.2. *类别*是一个有序集合 $\{S, K_1, \cdots, K_n\}$, 其中

(1) $n \geq 1$;

(2) 每个 K_i $(1 \leqq i \leqq n)$ 是知识对象或者类别, 称为这个类别的*成员*, 或称*属于*这个类别;

(3) S 是用于封装 $K_1, \cdots, K_n$ 所包含内容的概括或总结性信息的对象, 称为这个类别的*主题对象*.

若一个类别的成员是一个类别, 则它也可称为*子类别*.

类似于知识对象, 主题对象内部的数据结构通过*主题类*来刻画. 例如主题类包含类别的角色、名字、主题内容等属性, 而每个主题对象都拥有相应的属性值. 例如, 一个主题对象相应地拥有属性值“章”、“与一个三角形有关的点和线”、“本章的目的是介绍与

三角形相关的重要的点和线……”等. 为了在统一的框架下来建模领域知识的元知识, 可以使用知识图来表示类别. 主题对象与其成员之间构成了“包含”关系. 知识对象和主题对象是进行管理、处理以及检索等操作的基本单位, 可统称为*目标对象*.

基于以上的分析, 类别的结构可以用知识图 $(\mathscr{O}, \mathscr{I})$ 来表示, 其中 $\mathscr{O}$ 表示所有目标对象的集合, $\mathscr{I}$ 表示所有标签为包含 (include) 的链接的集合.

2.3.3　获取元知识

知识图通过详细描述知识对象之间的关系以及知识对象所属类别来结构化领域知识, 从而为元知识提供了表示方法. 因此, 根据知识管理的不同目的, 需要制定不同的知识图 (建模关系类型, 定义类别等).

元知识主要通过解决领域问题来获取, 并且随着领域知识的发展而扩充. 一般来讲, 推理与计算是解决问题的两种主要方式. 问题的解答 (证明或计算过程) 暗示了知识是怎样相互联系以及发展的. 目前有很多基于逻辑 (或人工智能) 方法的智能软件系统或工具, 包括定理证明器、证明助手、证明验证器、问题简化器、理论探索器等, 可以自动或半自动地帮助人们解决问题, 并自动获得元知识 [113]. 近来人们使用证明助手 Coq 所构造开发的几何知识库提供了大量关于几何知识的元知识. 现有的电子资源库中存在的交叉引用信息, 以及几何教科书等文献中的章节结构也是元知识的来源. 基于 Web 的共享机制可以被开发用于获取几何专家所掌握的未发表的 (元) 知识. 另一方面, 按照一定的规则分析几何知识数据, 开发相应的程序可以自动获得一部分元知识, 如定义对象之间的继承和依赖关系 (可参见第 4.4.3 节).

2.4　几何知识的管理过程

基于所构建的几何知识及其元知识数据, 按照数据库结构的设计原理和流程开发相应的模型, 应用数据库管理系统便可以实现几何知识库. 管理几何知识的接口是以知识库为基础的.

2.4.1　知识数据的操作与维护

随着几何学研究的不断发展, 几何知识会逐渐地积累和扩充, 因此需要对几何知识库进行更新, 主要包括增添新的知识数据, 删除或者修改已构建的知识数据. 知识库需要提供检索的功能来帮助人们有效地获取知识数据. 一种方式是利用元知识. 由于元知识

数据指明了领域知识数据的相互关系, 因此通过对元知识数据的操作, 即知识对象以及知识图的结构遍历可以满足人们对给定知识对象的相关知识数据或对象的获取. 事实上这并非通常意义上的检索, 因为不涉及排序, 模糊匹配等操作, 而仅仅是“选择”. 另一种方式则是基于几何知识数据本身进行检索. 一方面, 针对几何知识的自然语言表述, 可以使用信息技术中的相关方法, 通过建立索引等来实现匹配及排序; 另一方面, 针对使用某种形式语言来表述的几何知识, 因其有明确的语法规则, 因此可以利用这种语法规则来进行概念的精确匹配. 以此为基础, 研究几何构型的匹配方法 (目前并没有展开, 其过程将涉及到概念替换、简单的几何推理等一系列操作) 必然会为几何知识的检索提供有力的工具. 这些过程的实现主要依赖于所构建的知识数据在数据库中的存储结构.

另一方面, 由于几何知识库中的知识数据是对几何学相关事实的表述, 因此应当确保其在几何知识库的上下文中是正确的 (即知识库在逻辑上是一致的), 从而才能保证它们的使用是可靠的. 现有的证明助手 (如 Coq) 可以辅助人们来交互式地构造几何的形式证明, 尽管保证了几何定理及其证明在形式化的理论 (library) 上下文中是正确的, 但相关结果有限. 一些能力强大的几何定理机械化证明方法 (如坐标代数方法) 虽然不能产生可读证明, 也不能判断一个几何命题是否可以由知识库的上下文逻辑导出, 但可以判断其在通常语境中的正确性, 因此可以辅助人们检测知识库中部分几何知识的正确性. 由于知识数据的获取和构建大部分都是由人来完成的, 因此其正确性还是应当主要由其创建者来保证. 在数据库的更新过程中, 可能出现知识数据的冗余, 即相同的知识数据被重复构建. 虽然这并不影响其本身的正确性, 但是由于知识对象是管理操作的基本单位, 因此应当保证知识对象的唯一性, 尽量避免冗余的发生. 另外, 还可能出现知识数据不完备的情况, 即某些知识对象在逻辑上所依赖的知识对象并不存在于几何知识库中, 例如删除某个概念的定义使得此概念的应用变得没有依据, 或者修改某个公理使得公理系统发生变化而导致知识库在逻辑上变得不一致. 通过使用现有的软件工具来完全自动地进行上述检测是不可行的. 最有效的知识数据维护方式是人机协作, 即当进行知识数据更新时, 通过使用元知识来获取那些可能导致数据库逻辑上不一致、冗余以及不完备的知识数据, 然后由人对其进行分析判断再做出相应的改进操作.

2.4.2 知识数据的转换与交互

几何知识的使用者包括人和计算机. 针对不同的使用者, 几何知识需要通过不同的语言来表述, 即几何知识需要使用非形式的自然语言来表述以方便人的阅读和理解, 而

使用形式的语言方便计算机的访问处理与操作. 因此, 为了使用尽可能少的数据来表述几何知识就需要研究不同语言之间的翻译方法并开发相应的转化工具. 一般来讲, 将自然语言翻译为形式语言的难度依赖于形式语言的形式化程度, 而从形式化语言的表述转化为自然语言的表述则相对容易些, 但得到的结果可能对人而言并不那么自然. 这种转化需要使用自然语言处理的相关方法和技术. 另一方面, 由于不同的知识库使用的知识 (领域知识和元知识) 表述语言并不相同 (如 Coq 语言、Mizar 语言、OMDoc), 因此为了使知识在不同的知识管理系统中重用, 需要针对具体的语言分析研究不同知识库之间的数据转换方法. 为了整合和共享知识资源, 需要考察不同系统之间交互的可行性, 然后制定标准的知识表述语言, 这也需要各个机构、研究小组、用户等来协作完成 [62,110,117].

知识库中的知识数据需要呈现给人浏览和阅读, 因此需要设计漂亮的样式以及界面来全面生动地展示知识数据. 几何知识离不开几何构型, 动态几何软件可以通过动态的方式来呈现几何构型, 使得人们可以交互式地直观感受几何图形中所蕴含的性质. 另外 [133] 介绍了使用动态几何软件 Geometry Explorer 以图形的方式呈现几何证明的过程. 动态几何软件为呈现几何知识数据提供了有利的工具. 另一方面, 文字叙述的知识数据 (如自然语言的几何表述、代数方程等) 也是不可缺少的组成部分, 需要采用统一的字体、颜色、样式、交叉引用链接等来增加阅读的友好感受. 除了几何知识数据本身, 关于几何知识的元知识数据也需要呈现, 以便于人们从宏观的角度直观地了解几何知识的结构. 元知识包括知识对象的内部结构以及目标对象上的知识图, 前者需要根据相应知识类所定义的内部数据结构来设计具有层次结构的交互界面, 而后者可以利用具有交互功能的图的呈现工具来展示.

2.5　几何知识对象的计算、推理与可视化

如前所述, 在管理几何知识的过程中, 我们需要使用一些外部的软件工具辅助, 主要包括几何定理证明器或者计算机代数系统 (用于辅助判断几何命题的正确性、获取新的几何定理以及进行几何计算) 和动态几何软件 (用于动态地呈现几何构型).

为了能够应用这些工具, 需要将几何表述[注1]所使用的形式语言翻译成目标工具所

[注1] 这里的几何表述是指所要进行处理的数据, 如要进行定理证明时的几何定理的表述, 或者要呈现图形时的几何构型的表述.

使用 (或实现) 的语言. 对于实现而言, 需要将几何表述的表示格式转化为目标工具所识别的表示格式. 然而这种转化并不是简单的概念映射, 因为几何表述中会应用大量复杂的几何概念, 例如拿破仑三角形、高斯线等, 这些概念大部分都不是目标工具所使用的概念. 因此, 在应用这些工具之前, 先要对几何表述进行概念替换将其转化为另一个不使用这些概念的表述 (使用一些简单的概念如三角形、直线、交点、相等), 同时保证它们之间的等价性. 事实上, 概念替换也是处理问题的常用方式, 旨在依据概念的定义将一个概念用其定义另一端的表述代替, 从而将这个概念从原来的表述中“消去”. 几何知识库包含了几何概念定义的形式化表述, 为了使几何表述在几何知识库的上下文中被同义地转化, 需要研究使用这些陈述型的定义对几何表述进行转化的方法并将其算法化, 这一过程依赖于几何表述以及几何概念定义所使用的形式语言.

经过上述方法转化后的几何表述不含有复杂的几何概念, 因而可以被转化为目标工具可处理的表述, 这步转化并不是使用概念的定义, 而是使用概念上的映射. 事实上, 这种概念映射的规则是几何知识库中的过程化知识, 例如将“三点共线”转化为坐标代数方程的代数化规则; 将“以三点为顶点的三角形”转化为作图指令的图形化规则等. 因此, 需要进一步研究使用这些过程化知识来对几何表述进行转化的方法并将其算法化, 这一过程依赖于几何表述以及概念映射的规则所使用的形式语言.

一个几何表述经过以上的两步转化之后才可以被目标工具识别并进行相应的处理. 应用相似的方法, 形式化的几何表述也可以通过使用相应的过程化翻译规则被转化为自然语言表述. 因此转化过程的算法是应用外部软件工具对几何知识对象进行计算、推理以及可视化的关键.

第 三 章　几何知识库的设计与实现

3.1　引言

几何知识库是几何知识管理的研究重点, 也是几何知识管理系统的核心组成部分, 它为管理过程提供数据支持. 几何文献以及相关电子数据库所包含的几何定义、公理、定理、证明、问题等陈述型知识是几何知识库的主要构成部分, 也是管理的主要对象, 这些知识应当使用多种语言 (如英语、汉语、形式语言等) 来表述以满足不同使用者的需求. 从另一角度考虑, 目前已有很多用于自动推理、计算、交互式作图的几何软件工具, 每个软件都有其对相关几何知识 (主要是过程化知识) 的具体实现. 然而, 这些软件所使用的几何知识大部分都是相似的甚至是相同的, 例如, 在这些软件工具中, 一个圆可以通过三个不共线点, 两个点 (一个为圆心, 另一个为圆上的点), 或者一个点和一个长度 (作为半径) 三种方式确定地构造出来. 当开发新的几何软件工具时, 重新来详述及实现这些知识无疑会浪费大量的时间与精力. 因此建立一个可以为几何软件工具的开发以及使用提供标准知识数据的几何知识库, 将避免个体的重复劳动, 提高几何软件的实现效率.

几何知识是经过逻辑推理一步一步地创造和积累起来的, 例如利用已定义的几何概念引入新的几何概念, 应用已有的几何知识推导出几何概念的性质, 证明新的几何定理等. 几何知识并不是处在一个“平面”上, 而有其本质的层次结构. 几何知识库需要这些元知识来对几何知识数据进行结构化从而为管理操作提供支持, 全面准确地刻画元知识对于增加几何知识库的实用性起着不可或缺的重要作用. 研究与获取几何知识的元知识, 需要从以下三个方面考虑.

- 在教育中, 几何知识通常以文档的形式 (如教科书、讲义、论文等) 呈现给学生. 对于如何组织、表达及叙述知识, 教育工作者有很多使用惯例与共识. 文档的叙述结构依赖于所涉及知识之间的依赖关系. 因此为了辅助人们来创建结构合理的、可供人们学习的几何文档, 需要分析和整理几何知识的结构以便获取用于刻画几何文档中几何知识之间依赖关系的元知识;
- 在研究中, 需要追踪知识的发展轨迹: 某个概念是由哪些知识逻辑导出的, 哪些知识需要在证明某个定理中使用等. 因此, 为了提供给人一个清晰的几何知识的逻辑发展脉络, 需要获取几何知识之间逻辑关系的元知识;

- 在应用中, 元知识可以为几何知识数据的浏览、检索及维护提供支持, 因此需要针对应用的具体需求来获取相应的元知识.

*几何知识库*是用于存储和管理几何知识数据及其所属知识类和相互关系等元知识数据的特殊的数据库, 它需要提供几何知识的多种版本和不同形式的表述以满足创建电子几何文档和解决几何问题等自动化应用对知识数据的需求.

3.2　几何知识库的设计

下面, 我们应用第二章所提出的几何知识管理方法和技术讨论几何知识库的具体建造.

3.2.1　获取几何知识数据元素

数据元素是在知识库中存储知识数据的基本单元[注1]. 通过考察人与计算机两方面对知识库的使用需求, 我们分别从几何意义、代数意义、图形呈现三方面分析几何知识的表述, 进而获取几何知识数据元素.

概念为领域知识的陈述提供术语, 领域知识数据的基本组成是应用领域概念 (这里专指几何领域) 的具体实例. 例如, 三角形的内心是一个几何概念, 在一个具体的几何构型中, 它便专指一个具体的点, 这个点便是内心这个概念的具体实例. 在设计构建知识库时, 首先需要考虑的一个问题是如何检测几何知识的表述中几何概念的使用是否正确, 例如, 我们不能说一条直线和一个圆平行, 两条直线的切线. 为此, 几何知识需要使用某种形式语言来表述[注2]. 同时, 用于存储几何概念的使用规则的数据元素应该在知识库中被提供, 以便为概念的具体实例的使用进行正确性检测以及概念匹配.

- 一般来讲, 几何概念的具体实例是通过*构造*的方式表述的. 例如, 以三点为顶点的三角形, 三角形某边上的中线, 两条直线平行等. 一个几何概念需要 `vocabulary` 和 `attributeList` 两个数据元素来确定, 其中, `vocabulary` 元素用来存储区分概念的标识词汇, `attributeList` 元素用来存储概念的使用 (或构造) 规则. 例如, 三角形的概念可以用“triangle”[注3] 和“(Point, Point, Point)”表示, 用于指明三角形

[注1] 从技术上讲, 数据元素是用来描述数据的逻辑单元, 而数据域才是真正的存储单元. 为了叙述方便, 我们也称知识数据是存储在数据元素中的.

[注2] 这种形式语言的详细规范将在第 4.3.2 节中被讨论.

[注3] 引号中的部分表示数据元素所存储的数据.

需要由三个类型为 Point 的几何实体来构造. 在几何知识的表述中, 可以使用三角形的一个实例“triangle(A, B, C)”, 其中 A、B、C 必须是类型为 Point 的几何实体. 更复杂的实例如“triangle(midpoint(A, B), midpoint(A, C), midpoint(B, C))”表示三角形 ABC 的中点三角形, 这种表述是符合使用规则的, 因为“midpoint(A, B)”、“midpoint(A, C)”、“midpoint(B, C)”都是类型为 Point 的实体.

- 几何概念通常都是指在一般状态下的对象, 因而隐含着某些非退化条件. 例如, 一个三角形的隐含条件就是其三个顶点不共线, 否则这个三角形便退化成一条直线. 为了使概念的含义精确化, 需要数据元素 `nondegeneracyCondition` 来存储几何概念对应的非退化条件.
- 几何概念可以被理解, 需要其在几何意义上的定义. 从计算机的角度看, 需要用形式语言来表述其定义, 以便为几何表述的同义转化等应用提供支持, 因此知识库需要数据元素 `formalDefinition` 来存储概念的形式化定义; 从人的角度看, 需要以自然语言来进行表述, 以便直接将几何概念的含义呈现给人阅读以及理解, 因此知识库需要数据元素 `naturalRepresentation` 来存储几何概念的自然语言定义.
- 对几何概念的处理和使用还包括以下方面. 从代数意义来看, 几何概念都对应于代数中的元素, 例如, 点对应坐标 (或向量), 曲线对应代数方程, 平行垂直等几何关系对应代数方程, 因此知识库需要数据元素 `algebraicScript` 来存储将几何概念转化为相应代数形式的规则 (代数化规则), 从而为几何表述的代数化提供支持; 从图形呈现的角度来看, 几何概念对应于作图的构造命令, 例如, 以一点为圆心作过另一点的圆, 过某直线外一点作这条直线的垂线等, 因此知识库需要数据元素 `diagramScript` 来存储将几何概念转化为相应作图指令的规则 (图形化规则), 从而为几何表述的图形绘制提供支持; 从几何意义来看, 几何表述既需要形式语言也需要自然语言, 因此知识库需要用 `translationScript` 数据元素来存储将形式化表述的概念转化成自然语言表述的概念的规则 (翻译规则), 从而为形式语言翻译成自然语言提供支持.

下面考察几何知识的表述. 从计算机的角度看, 陈述型几何知识 (包括几何定理、证明、问题等) 需要使用具有精确语法规则的形式语言来表述, 使得计算机可以对其进行转化和处理, 因此知识库需要数据元素 `formalRepresentation` 来存储知识的形式化 (几何) 表述; 几何自动推理的最强大的方法就是将几何问题转化为相应的代数问题的

代数方法, 例如, 将几何定理的判定转化为 (半) 代数系统的理想成员判定问题, 因此知识库需要数据元素 algebraicRepresentation 来存储知识 (如定理、问题等) 的代数表述[注1]; 几何知识一般都对应于某个几何构型, 为了可以应用动态几何软件来呈现几何构型, 知识库需要数据元素 diagramInstruction 来存储构造相应几何构型的方法或作图表述[注2]. 类似于几何概念, 几何定理、命题等的成立也同样依赖于某些非退化条件的限制, 因此知识库需要数据元素 nondegeneracyCondition 来存储知识对应的非退化条件. 从人的角度考虑, 几何知识还需要被阅读和浏览, 因此知识库需要数据元素 naturalRepresentation 来存储相应的自然语言表述.

不论是几何概念的定义还是其它几何知识, 都需要被赋予一个名字作为标识, 例如, 概念的名字有三角形、内心等, 定理的名字有西门松定理、帕普斯定理等[注3], 因此知识库需要数据元素 knowledgeName 来存储知识的名字; 几何知识离不开几何图形, 通常教科书等文献中的几何图形是静态的, 即不可变化的, 其实质仅是几何构型的一个实例; 另一种几何图形是动态的, 即由动态几何软件绘制的, 图形在满足几何构型约束的条件下可以变化, 其实质是几何构型的所有实例, 而定理也正是要求对所有的实例都成立. 因此, 知识库需要数据元素 staticFigure 和 dynamicFigure 来分别存储静态图形和动态图形; 几何知识通常都具有一些关键的特征以区分彼此, 知识库需要数据元素 keyWords 来存储描述这些特征的关键字以方便查询和检索.

综上所述, 数据元素中所存储的知识数据可以服务于自动推理与计算 (基于代数化规则和代数表述)、自动作图 (基于图形化规则和作图表述)、自动翻译 (基于翻译规则)、同义转化 (基于概念的形式化定义)、不同版本的文档生成 (基于几何知识的不同语言表述、静态与动态图形) 以及检索 (基于名字、关键字). 关于各知识数据所使用的表述语言及格式, 我们将在第 4.3 节详细讨论.

3.2.2　构建几何知识对象

数据元素所存储的知识数据不是相互孤立的, 而是彼此之间存在着联系的. 例如, 同一个概念的定义可以使用不同自然语言来表述, 同一个定理在不同的坐标系统中可以有不同的代数表述等. 组织领域知识数据 (即建模元知识) 的第一个层次是将相互联系的数

注1 本文中的代数表述是指通过引入坐标系从代数意义上对表达对象的陈述.

注2 本文中的作图表述是指使用作图指令对几何构型的作图步骤的陈述.

注3 关于这两个定理的内容请参见附录 **A.**, 本文后面还将会使用到这两个定理作为例子.

据元素按照一定结构封装成可以被识别、区分、理解及操作的几何知识对象. 例如, 三角形内心的定义, 西门松定理等所涉及的数据元素都将被封装到相应的知识对象中. 知识类用来定义知识对象内部的数据结构, 即知识对象所包含的数据元素之间的联系, 数据元素即可看作是知识类的属性. 同一个知识类所定义的知识对象拥有相同的内部结构. 我们参考 OMDoc [75] 在陈述层次上的标注体系以及几何文献对知识内容的分类传统, 对几何知识对象进行如下分类, 并针对其特点详细分析定义在这些知识类的层次结构及其内部结构.

- 定义 (Definition): 用于赋予领域概念或术语以明确含义的知识;
- 公理 (Axiom): 领域内公认的真命题, 是推理的出发点, 并由此可以逻辑导出领域内真命题;
- 命题 (Proposition): 可以从逻辑上判断真假的领域知识;
  - 断言 (Assertion): 声明为真的命题;
  - 引理 (Lemma): 在证明定理过程中用作中间结论的真命题;
  - 定理 (Theorem): 由已接受的真命题, 如公理或引理等逻辑导出的真命题;
  - 推论 (Corollary): 由某个定理容易或直接地逻辑导出的真命题;
  - 猜想 (Conjecture): 还未被证明为真或为假的命题;
- 证明 (Proof): 说明命题为真的逻辑推导过程;
- 问题 (Problem): 关于某个数学对象或结构的疑问, 需要被解答或解释;
  - 例子 (Example): 用于引出或阐述领域知识所用到的问题;
  - 练习 (Exercise): 需要被解决的问题, 以便加强对某些特定领域知识的理解;
- 解答 (Solution): 解决问题的有限步过程;
- 方法 (Method): 用于解决问题的有限步策略;
- 算法 (Algorithm): 由输入得到结果的有限步操作过程;
- 说明 (Note): 用于介绍领域知识的目的、目标、背景、历史等的知识;
  - 介绍 (Introduction): 用于引入领域知识的概要说明;
  - 注释 (Remark): 对领域知识做出的某些注释说明.

每一个知识类只包含有特定的数据元素 (即属性), 具体的包含关系请参见附录 **B.** 下面讨论知识类所定义的知识对象的内部结构. 我们采用实体联系 (E–R) 模型来表示其内

部结构. 数据元素被视为实体, 我们首先通过分析其存储的知识数据的特征抽象出实体属性, 然后分析实体之间的相互联系.

一般情况下, 同一个知识对象包含的同一数据元素可能存储多种可能值. 例如, 一个概念定义的自然语言表述可能有多种, 一个定理可能有多种代数表述等等. 因此, 数据元素 algebraicScript、algebraicRepresentation、diagramScript 、diagramInstruction、nondegeneracyCondition 通过属性 version 来区分不同版本的知识数据; 数据元素 translationScript、naturalRepresentation 通过属性 language 来区分不同语言的知识数据;

而对于知识的名字, 不同语言的表述是不同的. 例如汉语用“西门松定理”, 英语则用“Simson′s theorem”. 因此我们对数据元素 knowledgeName 引入属性 name 来存储不同名字的表述, 如“西门松定理”和“Simson′s theorem”, 属性 language 用于存储名字所使用的语言名, 如“中文”和“english”, 属性 role 用于存储使用此名字的知识类名, 如“Theorem”.

几何概念的形式化定义一般由两部分组成: 待定义的概念与定义体语句. 我们由此对数据元素 formalDefinition 引入属性 instance 和 formalDefinition 来分别存储这两部分; 而对于陈述型几何知识的形式化表述通常由四部分组成: 前提数据 (如定理的假设、问题的前提、证明步骤中的前置条件等)、目标数据 (如定理的结论、问题的目标、证明步骤中的结论等)、数据关系 (如定理中假设与结论之间的充分或必要关系、问题中需要对目标进行证明、计算、作图或求轨迹、证明步骤中由前置条件得出结论的依据等)、序号 (如证明的过程即是对证明步骤的排列, 因此需要确定证明步骤在整个证明过程中的位置以便重构证明序列). 因此, 相应地我们对数据元素 formalRepresentation 引入属性 hypothesis、objective、relation、rank.

对于图形, 我们引入属性 name 用于存储其名字, dynamicFigure 来存储动态图形文件, staticFigure 用于存储静态图形文件, localPath 用于存储图形文件在本地存储设备上的路径.

对于几何概念, 存在着相同概念 (使用含义上相同的术语或名字) 对应多种构造的情形. 例如一个圆的概念可以有三种方式表达: *i*) 以一点为圆心, 过另一点的圆; *ii*) 以一点为圆心, 一个长度为半径的圆; *iii*) 过三点的圆. 这三种表述分别对应着数据元

素 attributeList 中存储的“(Point, Point)”,“(Point, Length)”,“(Point, Point, Point)”. 每种表述都决定了与此概念相关的其它数据元素所存储的知识数据的不同, 从而决定了不同的定义对象. 因此, 我们引入新的实体 concept 用于存储这些定义对象所拥有的共同信息以及实体 definitionObject 用于封装定义对象包含的数据元素. 同一概念可能会对应数据元素 vocabulary 中存储的多个值, 如圆的概念可以使用“圆”, “circle”或“c”作为其形式化的标识词汇; 同一定义对象可能对应数据元素 attributeList 中存储的多个值, 如过三点的圆可以使用“(Point, Point, Point)”或“(点, 点, 点)”作为其构造规则. 另外, 同一概念也可能对应数据元素 knowledgeName 中存储的多个值, 如圆的概念名字用中文是“圆”, 英文是“circle”, 德文是“Kreis”. 定义类的实体联系模型可参见图 3.1.

对于定义类以外的其它知识类, 同样存在着相同的知识 (含义上相同且属于相同的知识类) 对应多种版本的情形. 例如, 同一个定理的证明可能有不同的方式, 同一条定理存在着不同的形式化表述等. 每一种版本都决定了其所包含的知识数据的不同, 从而决定了不同的知识对象. 我们同样引入新的实体 knowledge 来存储这些知识对象所拥有的共同信息以及实体 knowledgeObject 用于封装知识对象包含的数据元素. 同一知识对象也可能对应数据元素 knowledgeName 中存储的多个值, 如西门松定理的中文名字是“西门松定理”, 英文是“Simson′s theorem”. 为了避免不同用户创建的知识对象之间的冲突, 我们为实体 concept 和 knowledge 引入属性 userName 来指明其创建的用户名以及属性 timeVersion 来存储其创建的时间. 其他知识类的实体联系模型可参见图 3.2.

3.2.3 构建知识图

知识对象将相互关联的领域知识数据组织成一个相对独立、语义明确的信息实体, 例如概念的定义, 定理等都是教科书中的基本知识点. 组织知识数据 (即建模元知识) 的第二个层次便是在知识对象的层次上分析获取其分类体系并建立相互关系.

知识对象上的分类体系

事实上, 我们在上一节引入的知识类可以看作是对几何知识对象的一种自然分类. 根据几何概念的性质和特征, 我们引入如下类别来对几何概念 (即定义对象) 进行分类.

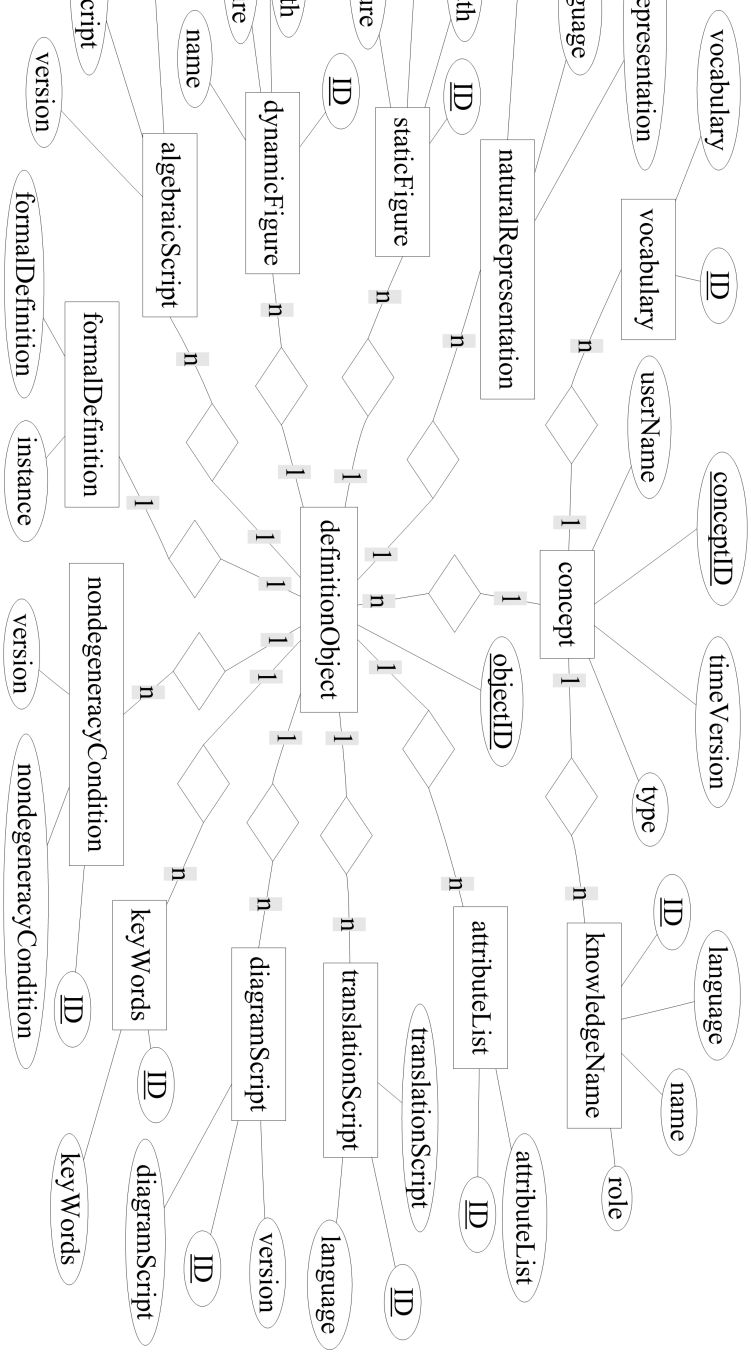
vocabulary
ID
naturalRepresentation
vocabulary
userName
conceptID
timeVersion
language
concept
type
ID
naturalRepresentation
objectID
knowledgeName
ID
language
name
role
localPath
ID
staticFigure
name
attributeList
attributeList
ID
attributeName
ID
staticFigure
definitionObject
translationScript
translationScript
ID
language
dynamicFigure
localPath
ID
diagramScript
version
ID
dynamicFigure
name
algebraicScript
ID
name
diagramScript
ID
algebraicScript
version
formalDefinition
keyWords
ID
keyWords
formalDefinition
instance
nondegeneracyCondition
ID
nondegeneracyCondition
version
1
n

图 3.1 定义类的 E-R 图

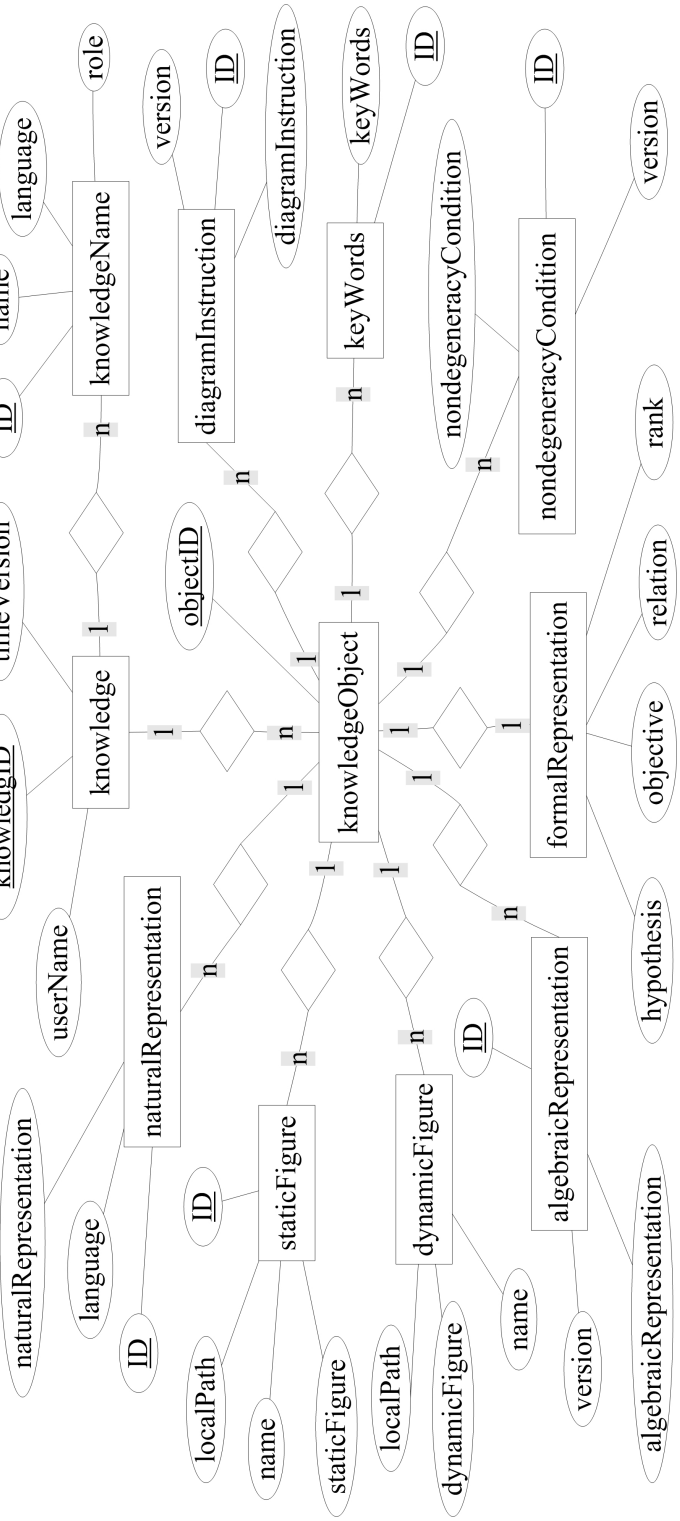

图 3.2 其它知识类的 E-R 图

- 几何实体 (包含如下子类别作为其成员)
  - 几何对象: 表示空间中几何体的几何概念, 如点、交点、直线、垂线、三角形、垂三角形、圆、内切圆、角、直角等;
  - 几何量: 表示几何对象的度量属性的几何概念, 如线段的长度、三角形的面积、圆的周长、角的大小等;
- 几何关系 (包含如下子类别作为其成员)
  - 几何对象关系: 表示空间中几何实体位置或大小关系的几何概念, 如两条直线平行、相交, 两个三角形相似等;
  - 几何量关系: 表示几何量大小关系的几何概念, 如两条线段的长度相等, 一个圆的面积比一个四边形的面积大, 一个角比另一个角小等.

对于其它类型的知识对象, 要进行合理而有效的分类并不容易. 一方面需要考虑知识本身的属性特征, 另一方面还需要考虑分类的应用场合. 根据命题的结论特征和性质, 我们对几何定理对象进行如下分类[注 1]:

- 共线: 如帕普斯定理;
- 共点: 如重心定理;
- 共二次曲线: 如帕斯卡定理的逆定理;
- 全等: 如全等三角形的判定定理;
- 相似: 如相似三角形的判定定理;
- 相等: 如毕达哥拉斯定理;
- 不相等: 如垂线段定理;
- 调和: 如三角形的调和性质;
- 平分: 如等腰三角形底边中线的性质;
- 平行: 如西门松对偶定理;
- 垂直: 如平行直线的性质;
- 共轭: 如三角形的共轭性质;
- 常量: 如等腰三角形的性质;
- 经过: 如圆过点的性质;

[注 1] 几何定理的具体内容请参见附录 **A.**

- 相切: 如费尔巴哈定理;
- 重合: 如 Nehring 定理.

另外, 通过人们的积累和总结, 几何文献中已存在大量的几何知识的分类信息, 例如, 用于阐述几何理论的教科书的各章节等都是对几何知识的分类, 都可以用于构建几何知识对象的分类体系.

分类体系指明了知识对象的分组或聚集特征, 对于知识导航、几何表述的形式语言设计以及几何文档生成等都起着重要的作用. 对于知识对象, 可以应用多个不同的分类体系对其进行组织. 为了使用统一的框架来组织知识对象并建模分类体系, 我们定义主题类内部的数据结构, 包含数据元素 `userName` 用于存储创建主题对象的用户名, `role` 用于存储主题对象的类型名 (如“章”、“节”), `name` 用于存储主题对象的名字 (如章的名字、节的名字), `title` 用于存储主题对象对其成员介绍的标题, `note` 用于存储主题对象对其成员介绍的内容, `timeVersion` 用于存储主题对象的创建时间. 由于主题对象所包含的数据可以使用不同自然语言来表述, 因此, 引入属性 `language` 用于存储数据所使用的语言, 不同语言决定了不同的主题对象.

知识对象的结构

分类体系是知识图的初步构建. 下面, 我们讨论知识对象之间可能存在的关系类型, 从而更深入细致地刻画知识对象的组织结构, 为知识图的全面构建提供条件.

根据几何知识之间的逻辑联系、教育的一般规律、教科书编写的传统规则等, 我们引入以下几何知识对象之间的关系类型并分析定义它们的层次结构和范围约束.

1. 概念 (定义对象) 之间的继承 (inherit) 关系. 一个新概念是通过使用已经定义的概念来引入的. 一个概念的定义包含两个部分: 一部分用来指明这个概念的父概念, 另一部分用来说明这个概念区别于其父概念的特别之处, 即一些几何约束条件. 例如, 线段的中点是一个特殊的点, 它的父概念是点的概念, 特殊的部分是这个点在这条线段上并且到线段两个端点的距离相等. 概念之间的这种表达范围上的包含关系称为*继承关系*, 概念拥有其父概念的性质. 通过概念的定义就可以得到继承关系.

   我们考察了欧几里德平面几何中的常用概念, 根据它们之间的继承关系构建出知识图 (如图 3.3). 其中节点表示几何概念对应的定义对象. 若 A、B 都为定义对象

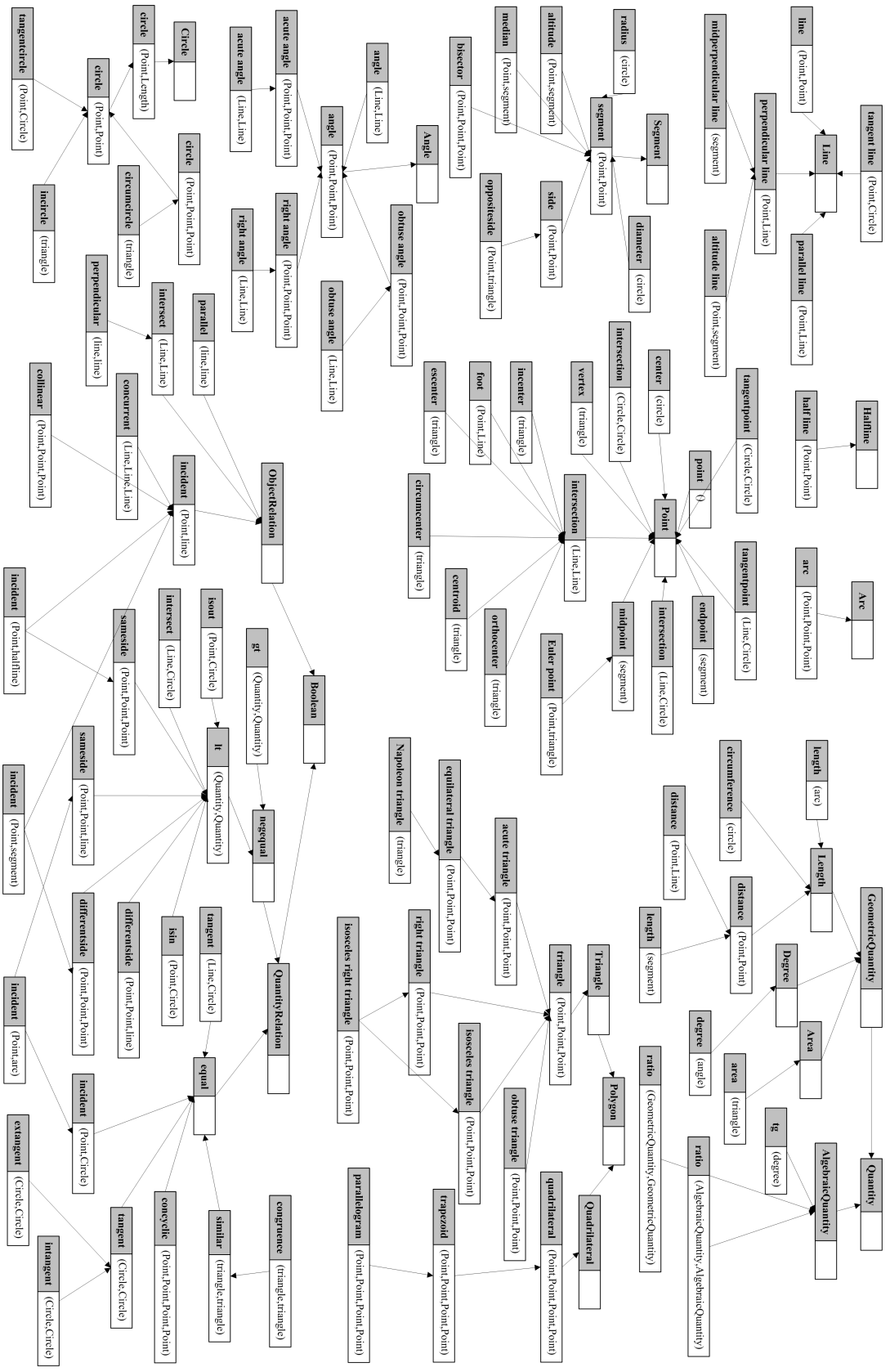
line (Point,Point)
tangent line (Point,Circle)
Line
perpendicular line (Point,Line)
parallel line (Point,Line)
midperpendicular line (segment)
altitude line (Point,segment)
half line (Point,Point)
Halfline
arc (Point,Point,Point)
Arc
radius (circle)
segment (Point,Point)
Segment
diameter (circle)
altitude (Point,segment)
median (Point,segment)
bisector (Point,Point,Point)
oppositeside (Point,triangle)
side (Point,Point)
tangentpoint (Circle,Circle)
intersection (Circle,Circle)
center (circle)
vertex (triangle)
incenter (triangle)
foot (Point,Line)
escenter (triangle)
point ()
Point
intersection (Line,Line)
circumcenter (triangle)
centroid (triangle)
orthocenter (triangle)
Euler point (Point,triangle)
midpoint (segment)
tangentpoint (Line,Circle)
intersection (Line,Circle)
endpoint (segment)
Angle
angle (Point,Point,Point)
angle (Line,Line)
acute angle (Point,Point,Point)
acute angle (Line,Line)
right angle (Point,Point,Point)
right angle (Line,Line)
obtuse angle (Point,Point,Point)
obtuse angle (Line,Line)
Circle
circle (Point,Length)
circle (Point,Point)
circle (Point,Point,Point)
circumcircle (triangle)
incircle (triangle)
tangentcircle (Point,Circle)
perpendicular (line,line)
parallel (line,line)
intersect (Line,Line)
collinear (Point,Point,Point)
concurrent (Line,Line,Line)
ObjectRelation
incident (Point,line)
Boolean
incident (Point,halfline)
sameside (Point,Point,Point)
isout (Point,Circle)
intersect (Line,Circle)
gt (Quantity,Quantity)
lt
(Quantity,Quantity)
negequal
sameside (Point,Point,line)
incident (Point,segment)
differentside (Point,Point,Point)
differentside (Point,Point,line)
isin (Point,Circle)
tangent (Line,Circle)
equal
QuantityRelation
incident (Point,arc)
incident (Point,Point,Circle)
extangent (Circle,Circle)
intangent (Circle,Circle)
tangent (Circle,Circle)
concyclic (Point,Point,Point,Point)
similar (triangle,triangle)
congruence (triangle,triangle)
length (arc)
Length
circumference (circle)
distance (Point,Line)
distance (Point,Point)
length (segment)
GeometricQuantity
Degree
Area
degree (angle)
area (triangle)
ratio (GeometricQuantity,GeometricQuantity)
tg (degree)
ratio (AlgebraicQuantity,AlgebraicQuantity)
AlgebraicQuantity
Quantity
Napoleon triangle (triangle)
equilateral triangle (Point,Point,Point)
right triangle (Point,Point,Point)
isosceles right triangle (Point,Point,Point)
acute triangle (Point,Point,Point)
isosceles triangle (Point,Point,Point)
triangle (Point,Point,Point)
Triangle
obtuse triangle (Point,Point,Point)
Polygon
parallelogram (Point,Point,Point,Point)
trapezoid (Point,Point,Point,Point)
quadrilateral (Point,Point,Point,Point)
Quadrilateral

图 3.3 继承关系知识图

且构成继承关系, 那么链接 $A \rightarrow B$ (标签为 `inherit` 且在图中省略) 表示 A 是 B 的一个特殊情形. 10 个本原概念 (即没有父概念的概念) 被选择作为根节点, 另外 11 个抽象概念被选作分支节点, 这些本原和抽象概念只能通过自然语言的表述来定义而没有形式化的精确定义. 尽管直观上, 直线 (Line) 并不是本原的而可以由点来定义, 但从几何计算与推理的角度看, 直线是最基本的构造. 同样圆 (Circle)和多边形 (Polygon) 也类似. 几何量 (GeometricQuantity), 例如长度、面积、角度, 和代数量 (AlgebraicQuantity) , 例如实数和复数, 都继承量 (Quantity). 对象关系 (ObjectRelation) 和量关系 (QuantityRelation) 都继承布尔量 (Boolean). 继承关系的知识图允许多继承的类树结构, 例如, 等腰直角三角形同时继承直角三角形和等腰三角形的性质.

2. 知识对象之间的依赖关系. 首先我们分析一下知识的积累与发展过程. 从一组基本概念和公理出发, 通过逻辑推理得到一些逻辑为真的命题, 成为定理. 为了叙述上的简洁并依赖于某些事实, 可以定义新的概念, 接着运用已知的概念和定理等知识通过逻辑推理, 又得到关于这些新概念的定理, 再重复这个过程. 同时, 通过提出一些问题 (或称为猜想, 通常解决起来很困难) 来指引知识的发展方向. 从这个简单的过程可以看出, 知识并不在一个水平层次上, 而具有一定的层次结构. 决定这种结构最主要的因素是先后顺序. 例如, 只有利用一些已定义的概念才可以定义新的概念, 并且定义中使用的几何约束条件不能相互矛盾; 只有预先定义了需要使用的概念, 才能应用这些概念来表述其它知识; 在证明几何定理的过程中, 会使用到已经证明过的定理; 在证明一个复杂定理之前, 先证明一些中间结果 (即引理) 是数学研究以及文档编排的手段, 从而使原证明过程不至于过长; 在一个复杂定理之后, 有时会给出其在特殊情形下的版本 (即推论), 而推论在实际中可能应用得更广泛. 知识对象之间的这种先后顺序关系称为*依赖关系*. 依赖关系也是在编排组织教科书内容时所应考虑的主要因素. 依赖关系包含如下关系类型:

   - 语境关系 (Context): 链接 $A \rightarrow_{\text{contextOf}} B$ 表示 A 提供 B 所需的语境, 其中 A 是定义对象, B 是知识对象;
   - 导出关系 (Derivation): 链接 $A \rightarrow_{\text{deriveFrom}} B$ 表示 A 由 B 推导得出, 其中 A 是定义对象, B 可以是公理、命题或问题对象;
   - 蕴含关系 (Implication): 链接 $A \rightarrow_{\text{imply}} B$ 表示 A 在 B 中被使用, 其中 A 可以

是公理、真命题、方法或算法对象, B 可以是证明或解答对象;

- 属性关系 (Property): 链接 $A \rightarrow_{\text{hasProperty}} B$ 表示 A 具有性质 B, 其中 A 是定义对象, B 可以是公理或真命题对象;
- 决定关系 (Decision): 链接 $A \rightarrow_{\text{decide}} B$ 表示 A 为 B 提供判定规则, 其中 A 是真命题对象, B 是定义对象;
- 引入关系 (Introduction): 链接 $A \rightarrow_{\text{introduce}} B$ 表示 A 引入或者介绍 B, 其中 A 是介绍对象, B 是主题对象或者除了说明对象以外的知识对象;
- 注释关系 (Remark): 链接 $A \rightarrow_{\text{remarkOn}} B$ 表示 A 为 B 作注释, 其中 A 是注释对象, B 是主题对象或者除了说明对象以外的知识对象;
- 复杂化关系 (Complication): 链接 $A \rightarrow_{\text{complicate}} B$ 表示 A 比 B 复杂, 其中 A 和 B 都是问题对象;
- 解决关系 (Solution): 链接 $A \rightarrow_{\text{solve}} B$ 表示 A 是 B 的解答, 其中 A 是解答对象, B 是问题对象;
- 练习关系 (Exercise): 链接 $A \rightarrow_{\text{exerciseOf}} B$ 表示 A 是一个与 B 相关的练习, 其中 A 是练习对象, B 可以是定义、命题、方法、算法或主题对象.

3. 知识对象之间的关联关系. 定理及其证明之间, 例子与目标之间, 等价知识之间通常是彼此对应的. 知识对象之间的这种彼此相联系的关系称为*关联关系*. 满足关联关系的知识对象在教科书等文献中通常一起出现并构成一个完整的部分[注1]. 我们也将满足关联关系的知识对象一起称作*知识块*.

   - 证明关系 (Justification): 链接 $A \rightarrow_{\text{justify}} B$ 表示 A 是 B 的证明, 其中 A 是证明对象, B 是真命题对象;
   - 应用关系 (Application): 链接 $A \rightarrow_{\text{applyOn}} B$ 表示 A 可以被应用于解决 B, 其中 A 是方法或算法对象, B 是问题对象;
   - 例子关系 (Example): 链接 $A \rightarrow_{\text{exampleOf}} B$ 表示 A 为 B 提供例子, 其中 A 是例子对象, B 可以是定义、命题、问题、方法或算法对象;
   - 联合关系 (Association): 链接 $A \rightarrow_{\text{associate}} B$ 表示 A 与 B 相互关联, 其中 A 与

[注1] 构成关联关系的两个对象之间并不会存在先后顺序. 例如可以先提出某个命题 (猜想) 再出现其证明, 也可以先存在某个证明, 再从中发现某个定理; 可以从某个例子引出某个命题, 也可以对某个定理附以应用实例作为例子. 依赖关系与关联关系之间的差别类似于序列与集合之间的差别.

B 可以是知识或主题对象;

- 等价关系 (Equality): 链接 $A \leftrightarrow_{\text{equal}} B$ 表示 A 与 B 有相同的含义, 其中 A 与 B 都是知识对象.

我们以 Geometry Revisited [35] 中的一节为例来展示基于以上关系类型的知识图 (见图 3.4). 图中的圆表示节点, 带有文本的箭头表示链接. C 表示主题对象, 其主题关于 Steiner-Lehmus 定理; T_1 表示 Steiner-Lehmus 定理对象; P_1 表示 T_1 的证明对象; D_1 表示角平分线的定义对象; T_2 表示定理对象“三角形的三条内角平分线共点”; D_2 表示定义对象“三条内角平分线的公共点被称为三角形的内心”; D_3 表示定义对象“三角形内接圆的圆心被称为三角形的内心”; L_1 和 L_2 表示在 P_1 中使用的两个引理对象; P_{11} 和 P_{12} 分别表示 L_1 和 L_2 的证明对象; I_1 是用于说明 T_1 的背景的介绍对象; R_1 是用于说明 T_1 的历史的注释对象; R_2 是 P_1 的注释对象; E_1 是与 T_1 相关的练习对象; E_2 是比 E_1 复杂的练习对象; S_1 和 S_2 分别是 E_1 和 E_2 的解答对象.

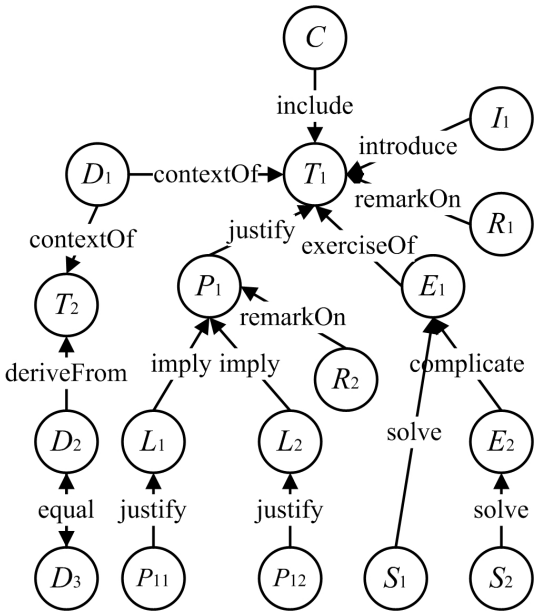

图 3.4 **Geometry Revisited 一书中第 1.5 节的知识图**

以上关系类型可以被用来详细描述知识对象或主题对象之间相互依存的特征. 关系类型是可以扩展的, 即可以在应用中定义所需的新关系类型. 事实上, 关系类型也有其属性特征以及相互关系. 目前我们仅给出这些关系类型上简单的分类体系, 即它们的层次结构, 而它们之间的相互关系有待进一步地研究和构建.

3.3　几何知识库的实现

3.3.1　构建关系表

根据上述对几何知识库的分析与设计, 我们实现了一个几何知识库, 用于存储和管理知识对象所封装的几何知识数据、主题对象所包含的信息以及这些目标对象上的知识图. 目前有很多数据管理技术可以用于实现这样相应的数据模式, 例如 XML 技术、关系数据模式等. 我们选用关系数据模式, 依据第 3.2.2 节所提出的知识类实体联系图以及主题类所包含的相关属性, 通过转化得到相应的数据关系表并构造了表间的依赖关系 (见图 3.5), 使用数据库管理系统 MS SQL Server 作为数据管理的平台.

在几何知识库中, 几何知识数据被封装成知识对象, 知识图构建于目标对象之上, 因此需要为每个目标对象赋予唯一的标识以便对其进行区分和指定. 为此, 我们制定了一个命名规则, 为每个目标对象赋予唯一的 `objectID`. 命名规则如下[注1]:

- 定义对象的 `conceptID` := `userName`.Definition.`knowledgeName`;
- 定义对象的 `objectID` := `conceptID`.`pattern`;
- 定义对象以外知识对象的 `knowledgeID` := `userName`.知识类名.`knowledgeName`; 其中知识类名可以是 Axiom、Lemma、Theorem、Corollary、Problem、Conjecture、Example、Exercise、Proof 或者 Solution;
- 定义对象以外知识对象的 `objectID` := `knowledgeID`.`version`;
- 主题对象的 `categoryID` := `userName`.Category.`role`.`name`;
- 主题对象的 `objectID` := `categoryID`.`language`.

基于以上的命名规则, 知识图中的节点可以通过引用目标对象的 `objectID` 值转化为可存储的形式. 一个链接 $A \rightarrow_\gamma B$ 本质上可由以下三元组来刻画.

- 前件 (precursor): 链接中的前件节点, 即 A;
- 后件 (subsequence): 链接中的后件节点, 即 B;
- 关系类型 (relationType): 链接中的标签, 即 γ.

[注1] 其中, `userName`、`knowledgeName`、`pattern`、`version`、`role`、`name`、`language` 分别表示目标对象包含的相应属性值.

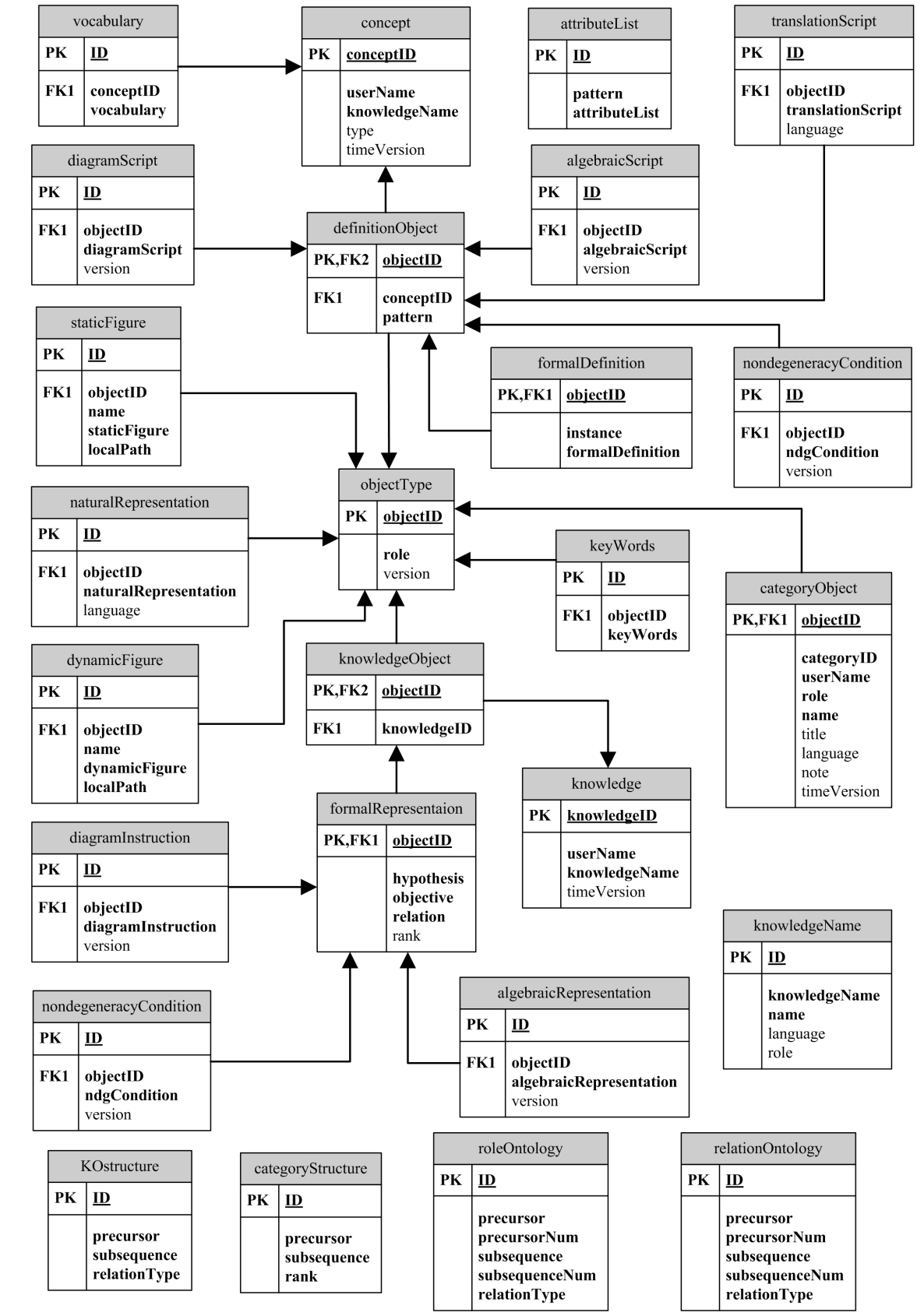
vocabulary
PK ID
FK1 conceptID
vocabulary
concept
PK conceptID
userName
knowledgeName
type
timeVersion
attributeList
PK ID
pattern
attributeList
translationScript
PK ID
FK1 objectID
translationScript
language
diagramScript
PK ID
FK1 objectID
diagramScript
version
definitionObject
PK,FK2 objectID
FK1 conceptID
pattern
algebraicScript
PK ID
FK1 objectID
algebraicScript
version
staticFigure
PK ID
FK1 objectID
name
staticFigure
localPath
formalDefinition
PK,FK1 objectID
instance
formalDefinition
nondegeneracyCondition
PK ID
FK1 objectID
ndgCondition
version
objectType
PK objectID
role
version
naturalRepresentation
PK ID
FK1 objectID
naturalRepresentation
language
keyWords
PK ID
FK1 objectID
keyWords
categoryObject
PK,FK1 objectID
categoryID
userName
role
name
title
language
note
timeVersion
dynamicFigure
PK ID
FK1 objectID
name
dynamicFigure
localPath
knowledgeObject
PK,FK2 objectID
FK1 knowledgeID
knowledge
PK knowledgeID
userName
knowledgeName
timeVersion
formalRepresentaion
PK,FK1 objectID
hypothesis
objective
relation
rank
diagramInstruction
PK ID
FK1 objectID
diagramInstruction
version
knowledgeName
PK ID
knowledgeName
name
language
role
nondegeneracyCondition
PK ID
FK1 objectID
ndgCondition
version
algebraicRepresentation
PK ID
FK1 objectID
algebraicRepresentation
version
KOstructure
PK ID
precursor
subsequence
relationType
categoryStructure
PK ID
precursor
subsequence
rank
roleOntology
PK ID
precursor
precursorNum
subsequence
subsequenceNum
relationType
relationOntology
PK ID
precursor
precursorNum
subsequence
subsequenceNum
relationType

图 3.5 数据存储结构

这样的三元组很容易通过构建具有相应属性的关系表存储在关系数据库中, 其中属性 precursor 和 subsequence 分别存储相应目标对象的 objectID 值, 属性 relationType 存储其标签. 因此, 知识图可以用数据库中一个关系表 KOstructure 的记录集合来表示. 由于标签为"包含" (include) 的链接是用来刻画知识对象的分类体系的, 我们单独为其创建一个关系表 categoryStructure. 由于一个类别所包含的成员之间可能存在着先后顺序, 例如教科书章节的排序, 因此我们为表 categoryStructure 引入属性 rank 来存储成员在其所属类别中的位置. 另一方面, 在目标对象的层次之上, 知识类之间以及知识图所使用的关系类型之间也分别存在着分类体系与相互关系. 为此, 我们在知识库中创建关系表 roleOntology 和 relationOntology 来分别存储它们的结构 (也可以看成是知识类和关系类型上的知识图, 见图 3.5).

3.3.2　管理数据

创建、修改、删除

根据各知识类以及主题类的内部数据结构, 我们设计并应用 Java 语言开发了相应的对话框及其与知识库的接口. 用户可以通过操作界面创建、修改、删除类型为定义、公理、引理、定理、推论、猜想、证明等知识对象和主题对象所包含的数据 (见图 3.6 和 3.7), 以及此对象与知识库中其它目标对象之间的关系 (即知识图中的链接), 从而实现了目标对象所封装的多层次数据以及元知识数据的输入和输出. 为了方便用户输入一些特殊格式的数据, 我们为知识库建立了与一些外部软件包的接口: 通过调用 Mathdox formula editor [91], 可以生成 OpenMath 语言编码的数学表达式; 通过调用动态几何软件 GeoGebra [46], 可以交互式地绘制动态几何图形并将其文件存储到数据库中.

当创建目标对象时, 接口将依据命名规则, 根据对话框中输入的数据自动生成相应的 objectID 值, 然后将数据保存到相应的关系表中; 而当修改目标对象时, 接口将通过用户提供的 objectID 值获取其对应的目标对象所包含的数据并将其显示到对话框中相应的位置, 然后将这些数据从知识库中删除, 当用户在对话框中修改完毕后, 新的数据将被保存到相应的关系表中; 当删除目标对象时, 接口做如下操作:

- 对于定义对象 (objectID 值为 id), 先从表 knowledgeName 中删除其 knowledgeName 值, 再将涉及到 id 的记录从表 KOstructure 中删除, 再将 id 从表

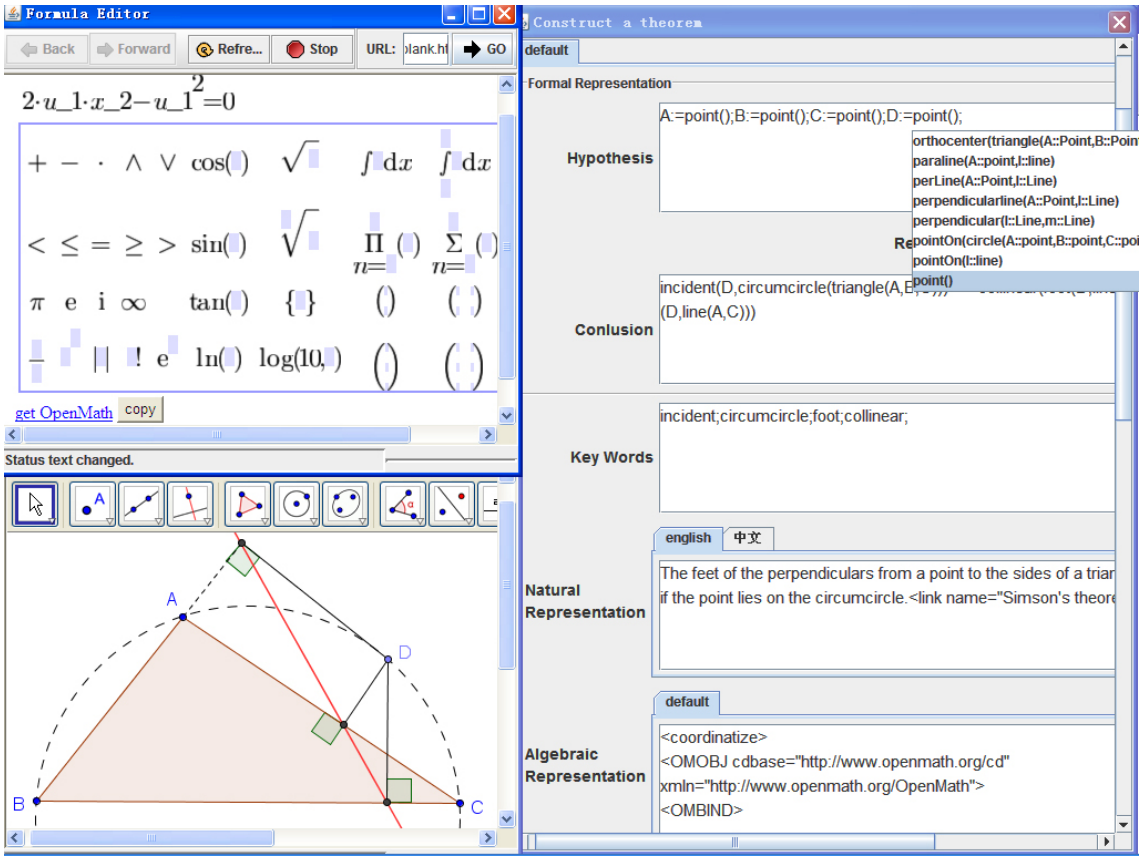

图 3.6 **创建 Simson′s theorem (西门松定理) 对象**

`objectType` 中删除, 最后将其 `conceptID` 值从表 `concept` 中删除;

- 对于其他知识对象 (`objectID` 值为 id), 先从表 `knowledgeName` 中删除其 `knowledgeName` 值, 再将涉及到 id 的记录从表 `KOstructure` 中删除, 再将 id 从表 `objectType` 中删除, 最后将其 `knowledgeID` 值从表 `knowledge` 中删除;
- 对于主题对象 (`objectID` 值为 id), 直接将 id 从表 `objectType` 中删除.

浏览与检索

几何知识库存储着几何知识对象所包含的知识数据、主题对象所包含的相关信息以及这些数据之间的相互关系, 需要以合适的粒度对其进行重新组织序列化并以可读文档的样式呈现给用户浏览阅读及使用. 我们考虑如下两种视角来呈现知识数据.

一种视角是浏览类别. 目标对象被呈现为树的节点, 主题对象的成员被呈现为其子节点. 我们将形如 $l(a_1, \cdots, a_n)$ 的表达式称为*逻辑表达式*, 其中 l 是逻辑连接词 and、or 或者 not, a_i $(1 \leq i \leq n)$ 是某个词汇或者是逻辑表达式. 命令 `browseBy[`R, N, U`]` (其

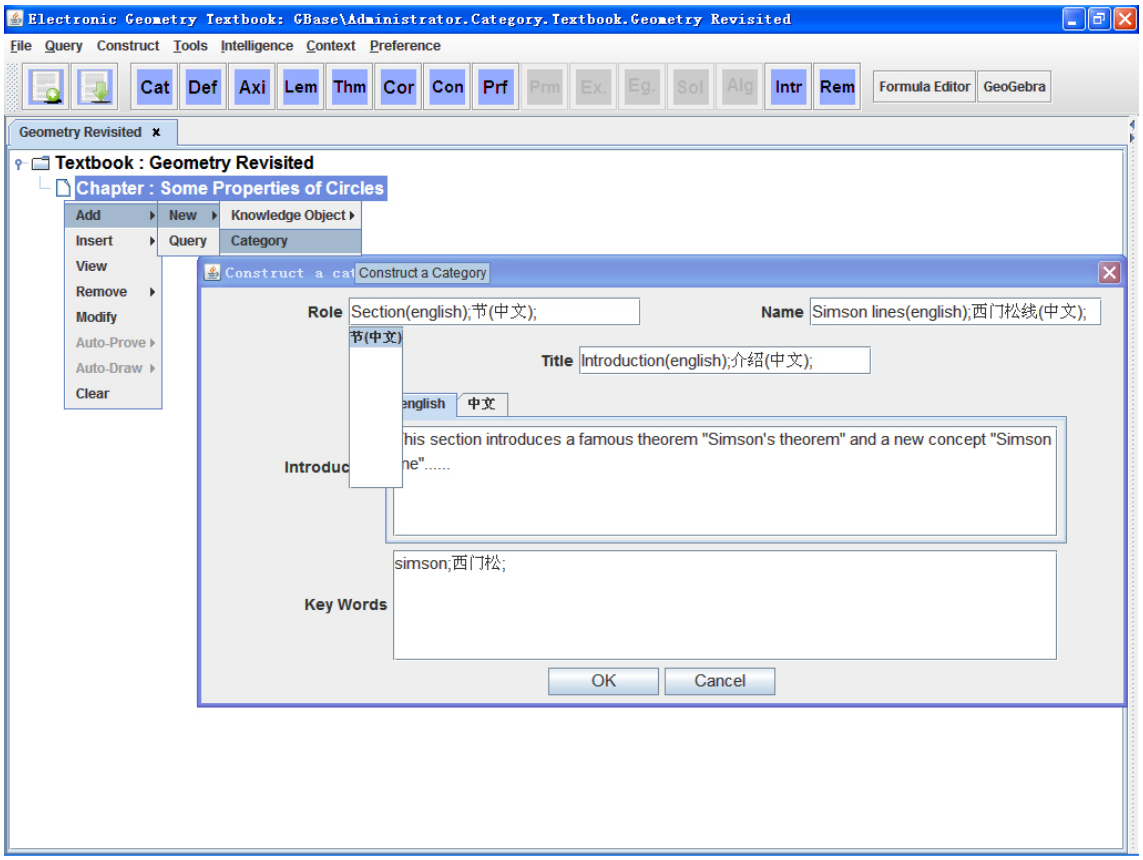

图 3.7 **创建名为“Simson lines (西门松线)”的节主题对象**

中 R 是一个知识类或者主题类的名字, N 是待浏览的目标对象的名字, U 表示用户名, 并且这三项都可以使用通配符“*”或者逻辑表达式) 表示浏览 U 所创建的类型为 R 名字为 N 的目标对象及其成员. 系统将首先依据第 3.3.1 节中的命名规则自动计算所浏览对象的 `objectID` 值并生成相应的节点. 若此对象为主题对象, 则从 `categoryStructure` 表 (见图 3.5) 中获取其所有成员对象的 `objectID` 值并生成相应的节点, 通过递归地执行相同的过程直到当前对象没有成员对象 (即当前对象为知识对象或者空主题对象[注1]) 时, 便完成了对类别的呈现. 按照这种方式, 每个节点都对应一个目标对象, 依据这些对象的 `objectID` 值, 系统通过检索 `knowledgeName` 表或者 `categoryObject` 表获得其对应的 `name` 属性值来作为节点呈现的名字.

另一种视角是浏览目标对象所包含的数据. 给定对象的 `objectID`值 (或者对象

[注1] 这里的空主题对象是指没有其它目标对象与其构成包含关系的主题对象. 由于没有成员, 按照类别的定义, 它不能构成一个类别. 但从数据的存储角度来看, 这样的空类别是可以存在的, 而且可以为其创建成员而使其成为一个类别.

对应的的节点), 系统根据数据存储结构 (见图 3.5) 来获取 `version` 值为“default”以及 `language` 值为当前系统语言 (“中文”或“english”) 的各种数据并自动将这些数据结构化为 XML 文档 (系统所使用的 XML 标签见附录 **C.**), 然后按照呈现需求开发 XSLT 样式表并使用 SAXON XSLT 处理器 [115] 将其转化为 XHTML 文档 (其中数学对象应用 MathML 表示), 最后通过 JDesktop Integration Components (JDIC [68], 提供了 Java 应用程序与本地桌面浏览器功能配置的接口) 来加载所生成的 XHTML 文档, 其中 MathML 通过 MathPlayer [94] 来呈现, 动态几何图形则通过 JavaScript 与 GeoGebraApplet 的接口 [47] 执行所获取的 `diagramInstruction` 值, 在 GeoGebraApplet 中呈现 (见图 3.8).

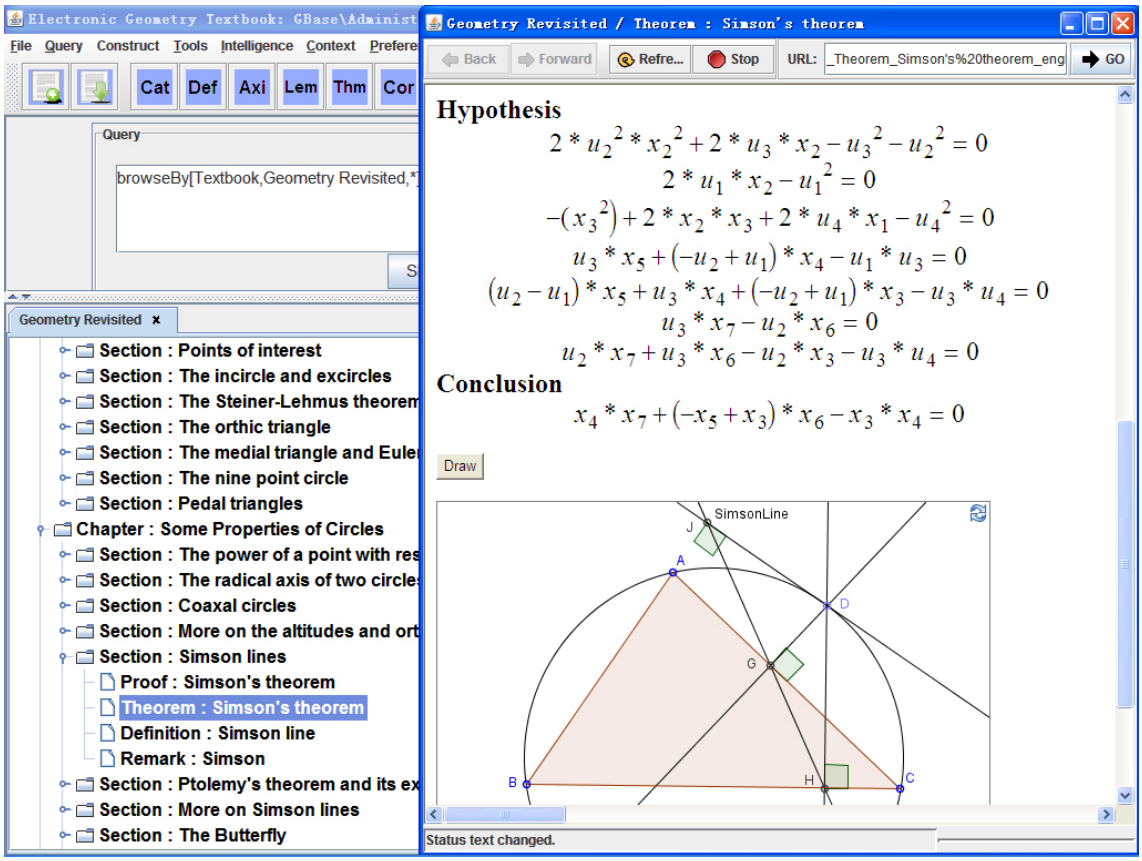

图 3.8 **浏览西门松定理对象所包含的知识数据**

知识检索是知识管理的一项主要任务. 它需要根据用户输入的检索需求描述, 输出符合要求的目标对象或其包含的知识数据. 因而需要将检索目标单位化. 我们将检索分为微观和宏观两种方式. 前者是指查找给定目标对象所包含的特定数据元素, 得到的结果是知识数据. 例如, 查找一个定理对象的形式化表述, 一个证明对象的图形, 一个定义

对象的中文表述等. 这种查找主要依赖于数据存储结构, 并已经在数据的创建、修改、删除接口中实现. 后者是指以目标对象为单位进行检索, 得到的结果是目标对象的集合.

根据数据存储结构 (见图 3.5), 系统实现了基本的宏观检索服务. 用户可以输入检索命令并得到相应的结果. 检索命令并不是基于知识的自然语言或者形式语言表述, 具体解释如下.

- `keyWords[`S`]` (其中 S 是一个逻辑表达式) 将返回 `keyWords` 表中 `keyWords` 属性值满足 S 的目标对象的集合;
- `relation[`$\star, S, T$`]` (其中 S 是一个逻辑表达式, T 可以是通配符"*"或者逻辑表达式) 返回 `KOstructure` 表中 `subsequence` 属性值满足 S 且 `relationType` 属性值满足 T 的目标对象的集合;

  `relation[`$S, \star, T$`]` (其中 S 是一个逻辑表达式, T 可以是通配符"*"或者逻辑表达式) 返回 `KOstructure` 表中 `precursor` 属性值满足 S 且 `relationType` 属性值满足 T 的目标对象的集合;

系统提供的检索命令支持用户获取与特定目标对象存在特定关系的目标对象的集合, 这对于几何文档的创建很有帮助. 例如, 如果用户想要将西门松定理包含到所创建的文档中, 他需要通过关键字来查询知识库中是否已经存在此定理的相关知识数据. 如果存在, 那么他就可以浏览其内容, 并修改数据以满足自己使用的要求. 另外, 为了确保文档的完备性, 需要将西门松定理所依赖的那些知识对象也包含在所创建的文档中, 例如垂足、共线、外接圆的定义都提供了西门松定理表述的语境. 由于检索的结果只是目标对象的集合, 为了采用合适的结构或者顺序来呈现结果, 我们采取浏览命令与检索命令相结合的方式来"样式化"结果集合. 命令 Q `and` B (其中 Q 是检索命令, B 是浏览命令) 表示在执行 B 所得的浏览结果中, 高亮显示那些同时存在于执行 Q 所得的结果集合中的目标对象所对应的节点 (见图 3.9).

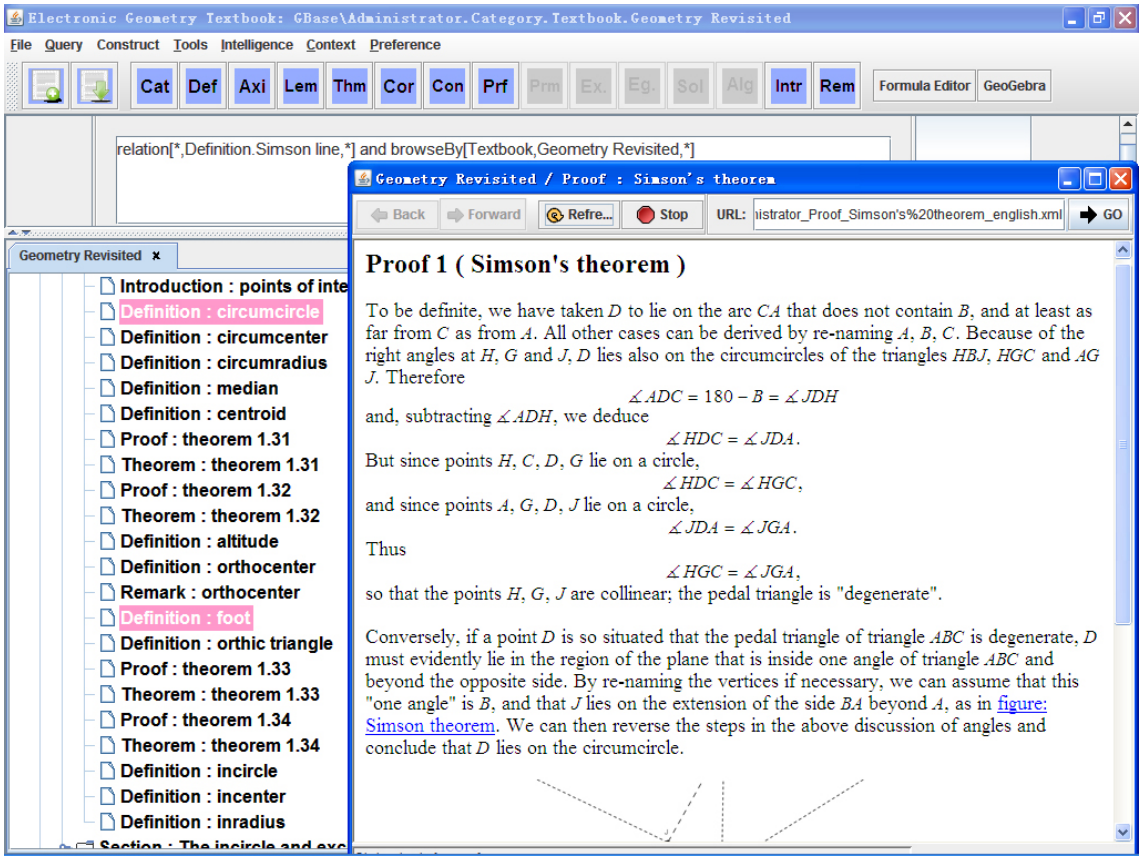

图 3.9 检索与西门松线定义有关系的前件对象并呈现西门松定理的证明

第 四 章　电子几何教科书系统的设计原理

4.1　电子几何教科书的目标和意义

教科书, 最常见的知识载体, 根据领域知识的逻辑结构有层次地阐述其内容和含义. 正是由于教科书对领域知识全面的记录以及合理的安排, 它们已经成为系统地存储、组织以及呈现领域知识的标准形式, 并在教育及研究领域起着十分重要的作用. 然而, 传统教科书都是以静态的文档模式来表示其内容, 通常仅关注知识的呈现而并没有考虑到对知识的其它管理功能. 这样的教课书在创建、维护以及使用上存在着不足, 具体表现在如下几方面.

- 教科书的创建存在大量重复性的工作. 例如, 常见的微积分教科书有很多种, 它们所涵盖的知识有很大一部分重合甚至连表述都相同, 这无疑浪费了人们的时间和精力;
- 随着领域的发展, 教科书需要不断地更新和改进, 教科书的内容也可能有多种不同的 (语言) 版本. 目前版本的维护过程并不是很有效. 当修改教科书某一部分内容时, 其它与这一部分有关的内容也需要作相应的修改, 但内容之间的这种关联并没有被有效地管理和充分地利用, 而只能由作者人工地维护;
- 教科书的内容必须是正确的或者说是逻辑上一致的, 并且是完整而没有重复的. 其叙述结构也要符合一定的教学规则, 以便更好地将领域知识呈现给读者阅读. 然而目前, 这些方面只能人为地检测;
- 教科书可以通过多种文本编辑排版工具来创建, 例如 Word、LaTeX 等. 然而, 这样的文档仅是计算机可读的, 而非计算机可理解的, 即所包含的知识并不是计算机可处理的, 所包含的问题并不能被计算机解决 (如证明定理, 构造动态的几何图形等). 这就使得教课书仅是“静态的”文档而不能与读者交互.

数学知识管理的思想和观点启发我们构想一种以动态的软件模式来表示和管理数学知识的新型教科书, 从而弥补静态文档模式教科书的不足. 基于几何知识管理的方法和技术, 我们提出了这样一个*电子几何教科书系统* (Electronic Geometry Textbook 或 EGT), 目的在于探索新的模式来整合现有的文献、软件和工具来辅助人们创建、维护与

共享多版本的、计算机可处理的电子几何教科书. 电子几何教科书系统的设计思想来源于以下三方面.

- 由于教学目的的不同, 相同的几何知识可能被不同的教科书采用. 另外, 几何知识包含多种形式, 其表述也可能具有不同版本以适用于不同应用. 例如, 内容的多种自然语言版本使得教科书可以在国际化的环境下使用, 知识的形式化表述版本将服务于自动知识处理过程. 为了共享和重用, 需要构建一个几何知识库来存储和组织多版本的几何知识数据. 电子几何教科书系统需要提供这样一个环境使得用户可以通过合适的粒度来管理教科书内容, 交互式地排列组织具有合适版本的知识数据来构建和维护几何教科书, 并且能够自动生成格式化的可读文档以便人们阅读浏览以及打印. 例如, 一本教科书可以被看成一系列节点的排列, 这些节点封装着相应的教科书内容, 如定义、定理、证明等. 用户可以执行一系列增加、插入、删除、修改以及重构这些节点的操作来构建、修改或者改进教科书, 并且同步地自动更新相应教科书文档的内容和结构;
- 在构建教科书的过程当中, 需要按照一个合适的叙述结构来安排教科书的内容. 尽管我们可以决定选取哪些知识, 但是在教学领域仍然存在着公认的组织、表示以及呈现领域知识的传统规则. 例如, 一个真命题仅当它被用于某个定理的证明过程当中才被认为是一个引理, 一个推论则一定是由某个定理逻辑导出的. 教科书应该按照领域知识的内在逻辑结构分层次地展开叙述, 即按照由简单到复杂循序渐进的顺序. 例如, 一段表述 (如定理、练习、例子等) 中所涉及到的概念的定义应该在这段表述之前被给出. 叙述结构上满足这些传统规则的教科书是结构一致的; 另外, 相同的内容不能重复地出现在教科书中, 即教科书是内容不冗余的; 所有必要的预备知识也需要在教科书中给出, 即教科书是内容完备的. 电子几何教科书需要能够实时地辅助用户分析所构建的教科书叙述结构是否一致, 并自动发现与传统规则不一致的部分. 我们称这个过程为教科书的*结构一致性检测*. 例如, 通常在一本教科书中, 三角形中线的概念只有当线段的中点被定义之后才可以被引入, 如果先引入中线的定义再引入中点的定义, 教科书的结构就是不一致的. 同样, 系统还需要辅助用户进行教科书内容的*完备性检测*与*冗余性检测*. 当所构建的教科书不满足这些约束时, 系统需要即时地将相关信息提示给用户以便改进和完善教科书的结构或者内容.

- 近年来, 几何学机械化方法的研究以及动态几何软件的开发都得到了很好的发展 (见第 1.2 节). 这些方法和工具不仅帮助几何研究者发现新的有价值的复杂几何定理, 而且也被广泛地应用于几何教育. 几何教课书通常包含有很多迷人的复杂定理, 但是这些定理的证明正确性仅可以由人检测. 使用定理自动证明器可以帮助作者检测书中的命题和证明问题是否成立, 尽管无法验证它们在教科书的上下文中是否是逻辑一致的, 但可以从通常意义上, 对其正确性作出辅助性的判断. 另一方面, 直观的几何图形是几何教课书不可缺少的部分. 应用动态几何软件可以动态地呈现几何构型, 从而增加教科书的交互性和探索性. 电子几何教科书系统需要提供 (形式化的) 知识数据与外部几何软件包的数据转化接口, 以使得教科书中的命题和证明问题可以被自动证明, 并可以自动绘制相应的动态几何图形[注1].

这种新型的电子几何教科书不仅可以象传统的 (静态文档) 教科书一样被浏览阅读和打印, 并且可以作为一个动态的软件在计算机上运行, 跟踪其内容的演化以逐渐增强其质量和适用性. 一些关于数学的网络学习和智能辅导系统, 例如 ActiveMath [3]、LeActiveMath [82]、MathDox [90], 可以根据不同用户的个性特征自适应地生成课件, 并且通过自适应的反馈来交互式地辅导学生, 分析评估学生的能力, 学习进度等并给予相应的学习建议. 然而这些系统是以学习者为中心的交互式学习系统. 电子几何教科书系统的设计目的主要是展示一种软件形式的动态的教科书. 系统的交互过程主要是使用者驱动的, 允许用户进行构建教科书的各种操作, 并由此通过定制合适的版本来不断地改进、完善和维护符合使用需求的教科书. 在这项研究中, 我们将构建大量的几何教科书等文献所包含的知识数据, 并以此为基础, 知识数据的维护、转化、呈现以及知识对象的计算、推理和可视化都将被深入地分析研究和实现, 从而有效地检测和实践几何知识管理的方法和技术.

4.2 系统构架与交互

下面, 我们描述系统的构架并阐述其工作原理. 图 4.1 所展示的是电子几何教科书系统的各个模块以及它们之间的交互. 教科书知识库是系统的核心组件, 用来存储与组织

[注1] 这里所指的自动绘制动态的几何图形是指由命题或证明问题的几何表述通过一系列操作自动生成其几何构型的作图指令 (而不是从知识库中的 diagramInstruction 数据元素直接获取), 然后在动态几何软件中执行这些指令进行呈现.

可共享的教科书知识数据. 通过用户界面, 用户可以调用操作模块来对知识库执行多种操作以构建及维护教科书, 例如创建新的知识数据, 检索需要的知识数据, 修改已有的知识数据等. 在此过程中, 检测模块将应用知识库中的元知识数据实时地检测当前教科书的完备性、冗余性以及结构一致性并将结果反馈回用户界面. 一方面, 所构建的教科书可以呈现为格式化的可读文档供人们浏览阅读及打印. 另一方面, 通过与外部几何软件包的接口, 教科书中所包含的命题和证明问题可以被自动证明, 并可以自动绘制相应的动态几何图形.

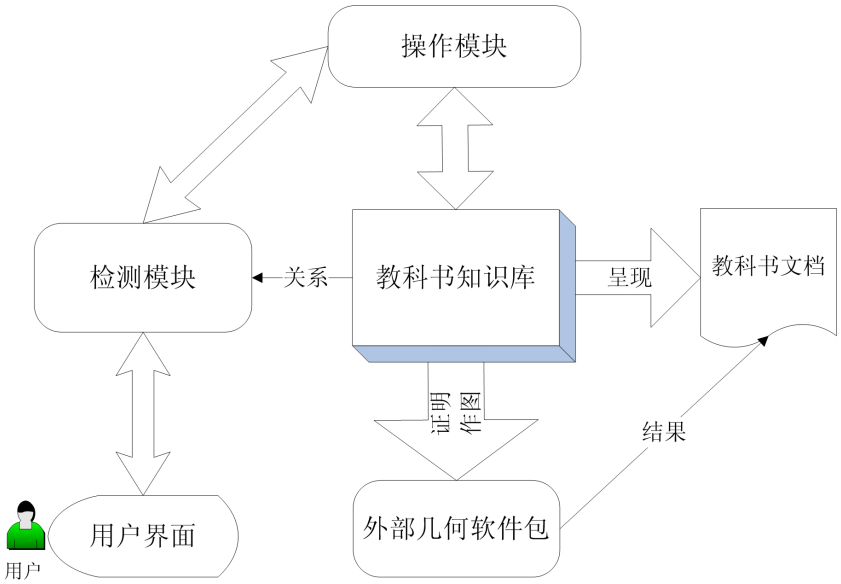

图 4.1 **电子几何教科书系统的构架**

从以上的系统描述, 我们可以看出这样一个系统是建于教科书知识库之上的, 各模块都是以知识数据作为操作对象. 教科书知识库事实上也是几何知识数据库, 因此我们利用第 3 章设计实现的几何知识库作为数据管理平台来存储、组织、维护、共享教科书包含的知识数据 (包括几何知识数据和元知识数据). 基于几何知识管理的方法与技术, 我们将研究内容分为两个方面: 一方面是详细描述教科书所包含几何知识的表述语言 (或者说是几何知识库中各数据元素所存储数据的规格), 尤其是设计几何表述的形式语言, 以便可以对其表述进行自动同义转化[注 1] 进而实现与外部几何软件工具交互; 另一方面集中讨论教科书的功能, 即基于教科书知识库, 设计与其它模块的接口.

[注 1] 本文的同义是指在几何概念定义的意义下, 所有的约束关系 (和构造方式) 都相同.

4.3 几何知识表述语言的设计

4.3.1 设计目标和意义

为了满足不同的应用需求, 几何知识需要通过不同的语言来表述, 如陈述型知识的几何表述所使用的自然语言, 代数表述所使用的 LaTeX 或者 OpenMath 语言, 作图表述所使用的动态几何软件作图指令语言等等. 几何教科书等文献所包含的几何知识是以陈述型知识为主体的, 并且形式是使用几何语言在几何意义上的几何表述[注1]. 为了使这些几何表述能够被计算机处理和操作以实现相关的自动化功能, 需要研究几何表述的形式语言.

概念是具有含义的认知单元, 是用于表述的基本单位. 特别地, 几何概念是用于表述几何命题、问题等各种知识的基本元素. 人们在几何知识积累的过程中不仅发现了定理, 而且发明了很多用于表述知识的几何概念. 几何概念通常是以构造的方式表达的. 一些概念通过直观的描述来定义, 这样的概念称为基本 (或简单) 概念, 例如过两点的直线、三点所成的角、以三点为顶点的三角形等. 导出 (或复杂) 概念通过应用基本概念和其他已定义的复杂概念来定义, 例如三角形的内心、拿破仑三角形等. 这些导出概念的应用使得几何表述更加简洁. 相同的几何知识可以应用不同的概念来表述.

以 Mechanical Geometry Theorem Proving [26] 一书中的 Example 198 为例, 通常在文献中表述为“The radius of the nine-point circle is equal to half the circumradius of the triangle”, 而为了可以更容易地代数化并使用坐标代数方法进行自动证明, 它可以表述为“Points A, B, C are arbitrarily chosen; $OA = OC$; $OA = OB$; D is on line BC; $DA \perp CB$; E is on line AC; $EB \perp CA$; F is on line AB; $FC \perp BA$; $NF = NE$; $NF = ND$. Then $2 * \overline{NF} = \overline{OA}$”. 这两种表述表达的是同一个定理的内容却应用了不同的概念. 后者使用的概念比前者更简单更易于代数化. “the nine-point circle of a triangle”和“the circumradius of a triangle”都是复杂概念而“is on line”和“=”都是简单概念. 通常人们使用的, 尤其是在几何教科书等文献中的几何表述都应用了大量的导出概念以使其更加简练, 方便交流.

[注1] 由于陈述型几何知识涉及到几何量的运算与操作, 因此几何表述不可避免地要涉及一些代数形式的表示, 但这种表示仍然是具有几何意义的. 我们使用几何表述的概念是为了区别于代数意义上的代数表述以及图形意义上的作图表述.

当使用已有的几何软件来证明定理, 绘制动态的几何图形时, 用户需要使用特定的工具来输入问题表述 (或者通过字符命令, 或者通过鼠标点击菜单按钮). 动态几何软件中使用的几何作图指令以及自动定理证明器中使用的几何约束谓词也可以被认为是几何概念, 例如 GeoGebra [46] 中的 Line[point A, point B] (过两点 A 与 B 的直线) 以及 GEOTHER [127] 中的 perpendicular(A, B, C, D) (直线 AB 与直线 CD 垂直). 大部分几何软件只可以识别 (或者说实现) 固定集合的几何概念 (通常都是基本概念), 即问题的描述只可以应用这个集合中的概念. 从解决几何问题的角度来看, 这个集合中的概念可以满足描述问题的需要. 然而从用户使用的角度看, 就需要先将问题的通常表述同义转化为几何软件可以识别操作的表述, 目前这个过程仅可以人为地完成. 这就给几何教科书等文献中几何定理证明与作图的自动化带来了阻碍. 尽管一些几何软件提供了宏扩展机制从而使用户可以自定义地扩充它们所能识别的几何概念, 但是这种扩展机制仅适用于软件本身, 同一个问题表述并不能在不同软件之间共享与重用. 要实现这种表述上的同义转化, 最直接的想法便是进行概念上的替换, 即通过导出概念的定义将原问题表述所应用的导出概念替换为简单概念. 事实上, 这也是我们在解决问题时通常使用的策略.

因此, 需要设计一个形式语言使得用户可以*自然*地、*容易*地、*简洁*地 (如允许嵌套表述, 可以应用集合概念等) 定义几何概念以及进行其它几何表述, 如表述几何定理、问题等. 并且需要研究使用这种语言的几何表述如何自动地同义转化为另一种易于处理的表述, 并通过实现特定的接口使得转化后的表述可以被几何软件识别处理. 我们设计了这样一种形式语言, 称为*几何描述语言* (*GDL*), 用户可以方便地表述几何概念、分句、构型以及定义、命题、问题等几何知识. 这样, 几何教科书中的几何表述通过使用这种语言形式化, 然后经过一系列的自动转化便可以实现与外部几何软件包的交互. 下面我们具体讨论这种语言的设计思想.

4.3.2　几何表述的形式语言

要设计几何表述的形式语言, 就需要考察研究几何表述的结构, 抽取出模式, 并针对不同的模式来定义相应的语法规则. 我们将几何表述按照语句组成的层次结构划分为如下部分.

概念

概念是几何表述所应用的有明确语义的基本单元. 由于几何学研究的对象不仅涉及

到空间中的几何实体, 而且还包括几何实体所具有的抽象属性, 即量. 因此, 几何表述还应用其它领域的概念, 例如代数学和集合理论中的概念. 由第 3.2.3 节的分析可知, 几何学本身所包含的概念, 即几何概念, 包含*几何对象*、*几何量*、*几何对象关系*、*几何量关系*.

代数概念则包含

- *代数量*: 表示抽象的数量, 如整数、实数等;
- *量函数*: 表示几何量或者代数量之间的运算操作, 如加法、乘法、正切、正旋等;
- *代数量关系*: 表示代数量之间的大小关系, 如等于、大于、小于等.

根据以上概念的语法属性, 我们将概念分为两类:

- *实体概念*: 表示名词性实体对象的概念, 包括几何对象、几何量、代数量、量函数;
- *布尔概念*: 表示可以判断真假的概念, 包括几何几何对象关系、几何量关系、代数量关系.

我们所考察的几何表述仅应用以上类型的概念. 这里, 我们将代数量与几何量区别开来是考虑到通常的几何量实质上是代数量与单位类型的有序对, 例如 $< q, \text{Length} >$ 表示长度 q, 与抽象的代数量 q 有着不同的含义. 由此, 我们将几何量关系与代数量关系区别开来, 例如长度 q 与代数量 q 相等是没有意义的, 尽管它们是关于同一个 q的量. 同样地, 面积 S 与长度 q 之间也不能比较大小.

为了使用统一的方式来形式化这些概念, 我们考察概念的结构. 很多概念都具有多态的特性, 即使用同一个词汇和不同的参数来表示不同的构造方式. 例如, 距离可以指两点之间的, 也可以指点到直线; 一个圆可以由三个点来构造, 也可以由一个点作为其圆心, 一个长度作为其半径来构造. 因此在应用时需要明确地区分不同的构造情形, 而不能使用单一的词汇. 通常, 概念都指明了构造的方式, 即由标识词汇和一列参数组成, 这些参数的目的是为了说明概念的使用 (或构造) 规则, 其构造方式是无关紧要的. 例如概念“三角形的垂心”, 指明了“垂心”是由“三角形”构造出来的, 即只要给定一个三角形, 而不论它在空间中位置、形状或者大小, 不论它的顶点是 ABC 还是 DEF, 都可以构造其垂心. 为了形式地表达概念, 我们引入*抽象实例*来表示一般化的虚拟实体, 例如对于上例的概念, 精确的说法是三角形 a 的垂心, 其中 a 便是一般化的虚拟实体, 可以是任何三角形. 抽象实例需要附加类型约束以规定其参数的取值范围, 例如上例中, “垂心”是标识词汇, a 为参数名, 且被约束为“三角形”这个类型 (不能是一个圆).

定义 4.3.1. 令 V 表示一个变量, 概念的形式表示如下:

(1) 形如 $V :: T$[注1] 的表述称为 *(抽象)实例*, 其中变量 V 称为此抽象实例的 *引用* (*reference*) 或 *指针* (*pointer*), 被赋予类型 T, T 称为 (*抽象*或*基本*) *类型*;

(2) 形如 $f(a_1, \cdots, a_n)$ $(0 \leqq n)$[注2] 的表述是一个 *概念*, 其中 f 表示几何或代数中使用的概念名字, 称为概念的 *词汇*, 每个 a_i $(1 \leqq i \leqq n)$ 都是抽象实例或者其它概念, 称为概念的 *参数*.

概念 C 含有"( )"的对数称为 C 的 *层数*, 记为 $\mathrm{L}(C)$. 概念只含有有限层数和有限个参数.

根据第 3.2.3 节对概念继承关系的讨论与分析, 抽象类型可以是 Point、Line、 Segment、Halfline、Angle、Circle、Arc、Triangle、Quadrilateral、Polygon、Length、Area、Degree、GeometricQuantity、AlgebraicQuantity、Quantity、equal、negequal、 ObjectRelation、QuantityRelation 或 Boolean. 这里所说的类型, 仅是符号上的标识, 是单一的词汇而没有具体的数据结构, 例如直线、圆、三角形等. 并且每个抽象类型都对应着某个本原或者抽象概念.

例如,

- A::Point 是一个抽象实例, 表示空间中一点 A;
- collinear(A::Point, B::Point, C::Point) 是一个几何关系, 表示 A、B、C 三点共线;
- circumcenter(triangle(A::Point, B::Point, C::Point)) 是一个几何对象, 表示三角形 ABC 的外心;
- distance(A::Point, l::Line) 是一个几何量, 表示点 A 到直线 l 的距离;
- distance(A::Point, B::Point) 是一个几何量, 表示两点 A、B 之间的距离.

分句

概念可以通过给定确切的或者具体的参数来实例化, 这样得到的表述称为概念的 *实例*, 也可以说该实例 *应用*了此概念. 例如, 假设已经构造空间中两点 D、E, 那么可以表述经过这两点的直线 DE, 它是概念"过点 A 和点 B 的直线"的实例. 值得注意的是点 A 和

[注1] 在一个形式表述中, 斜体标注的符号表示可以变化的量, 即可以用其它符号代替. 例如, A::Point 和 l::Line 都是形如 $V :: T$ 的表述. 而非斜体的符号则不能被替换.

[注2] 当 $n = 0$ 时, 形式为 $f()$.

点 B 是抽象实例, 它们可以代表空间中任意两个不同的点, 而点 D 和 E 是空间中已经存在的具体的点.

定义 4.3.2. *概念的实例*的形式表示如下:

(1) 一个变量, 例如 A、l 等, 是一个*项*;
(2) 一个常量, 例如 0、π 等, 是一个*项*;
(3) 形如 $f(a_1, \cdots, a_n)$ $(0 \leqq n)$ 的表述是一个概念的*实例*, 其中 f 是这个概念的词汇, 每个 a_i $(1 \leqq i \leqq n)$ 都是项, 抽象实例, 或者是其它概念的实例 (称为实例的*子实例*), 称为实例的*参数*.

实例 I 含有"( )"的对数称为 I 的*层数*, 记为 $\mathtt{L}(I)$. 概念的实例都只含有有限层数和有限个参数.

分句是在几何表述中说明事实的基本单位, 通过使用概念的实例来构造. 分句包含以下情形:

(1) 通过隐式的声明给定一般化的实体. 例如, "给定空间中的两个点 A、B", "l 和 m 是两条直线", 这里 A 和 B 分别指代任意的点, l 和 m 分别指代任意直线;
(2) 使用显式的引用来声明所构造的实体概念的实例. 例如, "令 l 是经过点 A 和点 B 的直线", 这里 l 指代经过点 A 和 B 的那条直线, A 和 B 指代已构造的点;
(3) 不使用显式的引用来声明所构造的实体概念的实例. 例如, "给定三角形 ABC", 这里没有使用一个名字来引用以 A、B、C 为顶点的这个三角形, A、B、C 指代已构造的点;
(4) 使用布尔概念的的实例, 例如, "直线 l 和直线 m 平行", 这里 l 和 m 指代已构造的直线.

定义 4.3.3. *分句*的形式表示如下:

(1) $\mathsf{declare}(\alpha_1, \cdots, \alpha_n)$ $(0 \leqq n)$ 是一个 (*声明*) *分句*, 其中每个 α_i $(1 \leqq i \leqq n)$ 都是抽象实例且 n 有限, 称为分句的*参数*;
(2) $V := I$ 是一个 (*引用*) *分句*, 其中 V 是一个变量, I 是一个实体概念的实例, V 被称为 I 的*引用* (*reference*) 或*指针* (*pointer*);

(3) and$(\alpha_1, \cdots, \alpha_n)$ 和 or$(\alpha_1, \cdots, \alpha_n)$ 是 (*引用*) *分句* (可称为*复合引用分句*), 其中每个 α_i $(1 \leqq i \leqq n)$ 都是引用分句且 n 有限, 称为分句的*参数*;

(4) give$(\alpha_1, \cdots, \alpha_n)$ 是一个 (*前提*) *分句*, 其中每个 α_i $(1 \leqq i \leqq n)$ 都是实体概念的实例且 n 有限, 称为分句的*参数*;

(5) 布尔概念的实例也可以被称为 (*布尔*) *分句*;

(6) 若 $\alpha, \alpha_1, \cdots, \alpha_n$ 都是布尔分句且 n 有限, 那么 and$(\alpha_1, \cdots, \alpha_n)$、or$(\alpha_1, \cdots, \alpha_n)$、not$(\alpha)$ 都是 (*布尔*) *分句* (可称为*复合布尔分句*), 其中 α, α_i $(1 \leqq i \leqq n)$ 称为分句的*参数*;

其中, (2) (3) (4) 又可以统称为 *(构造) 分句*.

例如,

- declare(A::Point, l::Line) 是一个声明分句, 意思是给定一点 A 和一条直线 l;
- $l :=$ line(A, B) 是一个引用语句, 意思是 l 为经过 A 和 B 的直线;
- give(triangle(A, B, C)) 是一个前提分句, 意思是给定三角形 ABC;
- perpendicular(l, m) 是一个布尔分句, 意思是 l 和 m 相互垂直.

构型或语句

我们将空间中的几何实体满足一系列几何关系或者量关系的几何表述称作*构型*或*语句*.

定义 4.3.4. configuration$(\alpha_1, \cdots, \alpha_n)$ $(0 \leqq n)$ 是一个*构型*, 其中每个 α_i $(1 \leqq i \leqq n)$ 都是分句或者构型, 称为构型的*参数*. 其中 n 有限并且所有变量或者是此构型中抽象实例的引用, 或者是引用分句中某个实例的引用, 同一个变量不能用作不同引用 (即引用不重名).

(1) 一个构型称为*构造构型*, 若每个 α_i $(1 \leqq i \leqq n)$ 都是声明分句, 或者构造分句, 或者也是一个构造构型;

(2) 一个构型称为*约束构型*, 若每个 α_i $(1 \leqq i \leqq n)$ 都是声明分句, 或者布尔分句, 或者也是一个约束构型;

(3) 其它的构型称为*混合构型*.

两个构型或分句可以合并. 若 $\mathcal{S}_1$ 为一个形如 configuration$(\alpha_1,\cdots,\alpha_n)$ 的构型, $\mathcal{S}_2$ 为一个形如 configuration$(\beta_1,\cdots,\beta_m)$ 的构型, 则 $\mathcal{S}_1$ 与 $\mathcal{S}_2$ 可以合并为一个构型 configuration$(\alpha_1,\cdots,\alpha_n,\beta_1,\cdots,\beta_m)$, 记为 $\mathcal{S}_1$+$\mathcal{S}_2$, 和另一个构型 configuration$(\beta_1,\cdots,\beta_m,\alpha_1,\cdots,\alpha_n)$, 记为 $\mathcal{S}_2+\mathcal{S}_1$.

例如,

- 西门松定理 (见附录 **A.**) 的几何构型可以表示为 configuration(A := point(), B := point(), C := point(), D := pointon(circle(A,B,C)), J := foot(D, line(A,B)), H := foot(D, line(B,C)), G := foot(D, line(A,C)));
- 几何构型 configuration(A := point(), B := point(), C := point(), give(ninepointcircle (triangle(A,B,C))) 表示 A、B、C 是任意三点, 给定三角形 ABC 的九点圆.

定义

几何概念定义的一般结构[注1]是"概念 A 被定义为 R 并满足约束 C", 其中 A 是待定义的概念, R 表示 A 的父概念, C 表示 R 应满足的用于限制 A 的范围的约束或构造条件. 例如, 两条直线的交点是一个点并且满足这个点同时在这两条直线上. 另一方面, 通常所定义的几何概念都是处于一般的情形, 非退化条件并没有明确说明, 例如, 三角形的概念隐含三角形的三个顶点不在同一条直线上.

定义 4.3.5. *定义的形式表示如下:*

(1) 形如 [I where $\mathbb{C}$] 的表述 $\mathcal{S}$ 是一个*约束绑定*, 其中 I 是一个抽象实例, 或者概念的实例, 或者布尔分句, 或者构型, 称为 $\mathcal{S}$ 的*中心实例*, 记为 CI($\mathcal{S}$); $\mathbb{C}$ 是一个布尔分句, 或者引用分句, 或者为 null[注2], 称为 $\mathcal{S}$ 的*约束*, 记为 Constraint($\mathcal{S}$);

(2) 形如 [V] 的表述 $\mathcal{S}$ 是一个*逆绑定*, 其中 V 是一个变量, 称为 $\mathcal{S}$ 的*中心实例*, 记为 CI($\mathcal{S}$);

(3) 形如 Definition$(C,\mathbb{R},\mathbb{N})$ 的表述 $\mathbb{D}$ 是一个定义, 其中 C 是一个概念, 称为 $\mathbb{D}$ 的*目标概念*, 记为 TC($\mathbb{D}$); $\mathbb{R}$ 是一个约束绑定, 或者逆绑定, 或者抽象实例, 称为 $\mathbb{D}$ 的*返回*

[注1] 由于我们关注于几何表述的形式语言, 因此这里只讨论几何概念的定义结构, 而代数概念的定义则不在本文的讨论范围之内.

[注2] 本文中的 null 表示空, 对 null 的任何操作都为空操作, 即操作将终止.

体, 记为 RB($\mathbb{D}$); $\mathbb{N}$ 是一个布尔分句或者 null, 称为 C 的*非退化条件*, 记为 NC(C); 并且满足以下要求:

- $\mathbb{C}$ 中出现的变量在 C 或者 I 中出现;
- 逆绑定的中心实例是 C 中某个抽象实例的引用;
- $\mathbb{N}$ 中出现的变量在 C 中出现;
- 同一个变量不能用作不同引用;
- 若 C 没有参数, 那么 $\mathbb{R}$ 是一个抽象实例.

若定义的返回体为抽象实例, 那么定义的目标概念称为*基本* (或*简单*) *概念*[注1], 否则称为*导出* (或*复杂*) *概念*. 若定义的返回体 $\mathbb{R}$ 为约束绑定, 那么当 Constraint($\mathbb{R}$) 为布尔分句或者 null 时, 此定义称为*约束型定义*; 当 Constraint($\mathbb{R}$) 为引用分句或者 null 时, 此定义称为*构造型定义*.

例如,

- Definition(intersection(l::Line, m::Line), [A::Point where and(incident(A, l), incident(A, m))], not(parallel(l, m))) 表示两条直线 l 和 m 的交点被定义为一点 A 满足 A 既在 l 上又在 m 上, 其非退化条件是这两条直线 l 和 m 不平行;
- Definition(completequadrilateral(A::Point, B::Point, C::Point, D::Point, E::Point, F::Point), [configuration(E := intersection(line(A, B), line(C, D)), F := intersection (line(A, C), line(B, D)))], null) 表示全四边形被定义为一个构型: E 为直线 AB 与直线 CD 的交点, F 为直线 AC 与直线 BD 的交点, 没有非退化条件.

定义 4.3.6. *脚本*是定义的集合, 形式化表示为 Script($\mathbb{D}_1, \cdots, \mathbb{D}_n$) ($0 \leqq n$), 其中每个 $\mathbb{D}_i$ ($1 \leqq i \leqq n$) 都是定义并且 n 有限.

命题

命题的一般结构是二元组 $(\mathcal{H}, \mathcal{G})$, 其中 $\mathcal{H}$ 是命题的前提部分, $\mathcal{G}$ 是命题的结论部分.

定义 4.3.7. *命题*的形式表示为 Proposition($N, T, \mathcal{H}, \mathcal{G}$), 其中 N 是命题的名字, T 是命题的类型名, 即 Assertion (断言)、Lemma (引理)、Theoerem (定理)、 Corollary (推论)

[注1] 几何表述中所出现的代数概念都被视为基本概念.

或 Conjecture (猜想), $\mathcal{H}$ 是一个几何构型, 称为命题的*前提*, $\mathcal{G}$ 是一个布尔分句或者形如 $\alpha_1 <=> \alpha_2$ (其中 α_1、α_2 均为布尔分句), 称为命题的*结论*. 结论中的变量都在前提中出现.

例如,

- 西门松定理 (见附录 **A.**) 可形式化为 Proposition(Simson's theorem, Theorem, configuration(A := point(), B := point(), C := point(), D := point()), incident(D, circumcircle(triangle(A, B, C))) <=> collinear(foot(D, line(A, B)), foot(D, line(A, C)), foot(D, line(B, C))));
- 帕普斯定理 (见附录 **A.**) 可形式化为 Proposition(Pappus, Theorem, configuration(declare(C::Point, F::Point, P::Point, Q::Point, R::Point), A := point(), B := point(), D := point(), E := point(), give(Pappus($A, B, C, D, E, F, P, Q, R$))), collinear($P, Q, R$)).

问题

问题的表述结构与命题的类似, 也是二元组 $(\mathcal{H}, \mathcal{O})$, 其中 $\mathcal{H}$ 是问题的前提部分, 而 $\mathcal{O}$ 是问题的目标部分.

定义 4.3.8. *问题*的形式表示为 Problem$(N, T, \mathcal{H}, \mathcal{O})$, 其中 N 是问题的名字, T 是问题的类型名, 即 Compute (计算)、Prove (证明)、Locus (求轨迹) 或 Construct (作图), $\mathcal{H}$ 是一个几何构型, 称为问题的*前提*, $\mathcal{O}$ 称为问题的*目标*. 根据不同的问题类型, $\mathcal{O}$ 有如下不同的形式:

(1) 若 T 为 Compute 或者 Locus 或者 Construct, 则 $\mathcal{O}$ 是一个实体概念的实例或其引用;
(2) 若 T 为 Prove, 则 $\mathcal{O}$ 是一个布尔分句或者形如 $\alpha_1 <=> \alpha_2$ (其中 α_1、α_2 均为布尔分句);

目标中的变量都在前提中出现.

例如,

- 计算面积问题“给定一个三角形, 一条边长为 a, 其上的高长为 h, 求这个三角形的面积”可形式化为 Problem(Area, Compute, configuration(declare(A::Point, B::Point, C::Point), give(triangle(A, B, C)), a := length(side(A, B)), h := height(C, side(A, B))), area(triangle (A, B, C)));
- 证明问题“平行四边形的对角线互相平分”可形式化为 Problem(Parallelogram, Prove, configuration(declare(D::Point), A := point(), B := point(), C := point(), give(parallelogram(A, B, C, D)), O := intersection(line(A, C), line(B, D))), and(equal(length(segment(O, A)), length(segment(O, C))), equal(length(segment (O, B)), length(segment(O, D))))).

含有集合的表述

在几何表述中, 通常会遇到多个对象统一表述的情形:

(1) 在使用显式的引用来声明实体概念的实例时, 可以统一表述. 例如, X、Y、Z 分别是三角形 ABC 三边 AB、BC、AC 的中点, m、n 分别是三角形 ABC 边 BC、AB 上的高;

(2) 当多个实例都应用相同的概念时, 可以统一表述. 例如, 过 A、B、C 三点分别与直线 a、b、c 垂直的直线;

(3) 有些概念包含多个对象, 并且这些对象满足相同的约束条件或构造方式, 在定义中需要枚举所有情形. 例如, 四边形的 2 条对角线, 直线与圆的 2 个交点, 三角形的 3 个顶点, 三角形的 3 条中线等;

(4) 有些概念虽然包含多个对象, 但这些对象存在的约束条件并不相同, 即概念在不同的情形下对应不同的对象, 在定义中需要区分这些情形所返回的对象. 例如, 一个圆的切圆包括内切圆和外切圆.

综合考虑以上情形, 我们对几何描述语言作如下扩展, 以增强其自然的表达能力.

(1) $\{V_1; \cdots ; V_n\} := \{\alpha_1; \cdots ; \alpha_n\}$ $(1 \leqq n)$ 是一个 (集合) 分句, 若每个 $V_i := \alpha_i$ $(1 \leqq i \leqq n)$ 都是引用分句且 n 有限;

(2) $\{V_1; \cdots ; V_n\} := I$ $(1 \leqq n)$ 是一个 (集合) 分句, 其中每个 V_i $(1 \leqq i \leqq n)$ 都是变量且 n 有限, I 是实体概念的实例;

(3) $f(\{\alpha_1;\cdots;\alpha_n\},\cdots,\{\beta_1;\cdots;\beta_n\})$ $(1 \leqq n)$ 是一个概念的 (集合) *实例*, 若每个 $f(\alpha_i,\cdots,\beta_i)$ $(1 \leqq i \leqq n)$ 都是概念的实例;

(4) $\{\mathbb{R}_1;\cdots;\mathbb{R}_n\}$ $(2 \leqq n)$ 是一个定义的 (集合) *返回体*, 若每个 $\mathbb{R}_i$ $(1 \leqq i \leqq n)$ 都是定义的返回体且 n 有限; 若每个 $\mathbb{R}_i$ $(1 \leqq i \leqq n)$ 都是约束绑定, 其中心实例都是某个概念的实例并且约束都为 null, 则这样的返回体称为*互斥集合返回体*, 否则称为*共存集合返回体*.

形如 $\{\alpha_1;\cdots;\alpha_n\}$ $(1 \leqq n)$ 的表述称为*集合*, α_i $(1 \leqq i \leqq n)$ 称为*分量*, 并且分量的个数 n 是有限的.

例如,

- $\{A;B;C\}$:= {point(); point(); point()} 是一个 (集合) 分句, 表示 A、B、C 是任意三点;
- $\{D;E;F\}$:= vertex(Napoleontriangle(triangle(A,B,C))) 是一个 (集合) 分句, 表示 D、E、F 是三角形 ABC 的拿破仑三角形的三个顶点;
- collinear(midpoint(diagonal(completequadrilateral(A,B,C,D,E,F)))) 是一个 (集合) 实例, 表示全四边形 $ABCDEF$ 的三条对角线的中点共线;
- Definition(diagonal(completequadrilateral(A::Point, B::Point, C::Point, D::Point, E::Point, F::Point)), {[segment(A,D)]; [segment(B,C)]; [segment(E,F)]}, null) 表示全四边形 $ABCDEF$ 有三条对角线, 分别是线段 AD、线段 BC、线段 EF.

另外, 在几何表述中还存在关于集合的操作概念. 例如, 三角形三个顶点中的任意一顶点, 三角形的任两条高线等. 我们在几何描述语言中引入如下形式, 以增强其表达能力.

choosediff($\{A_1;\cdots;A_n\},m$) $(1 \leqq m \leqq n)$ 是一个集合的 *(操作) 实例*, 表示从 A_i $(1 \leqq i \leqq n)$ 中任选 m 个不同的分量所组成的集合. 例如, line(choosediff($\{A;B;C\},2$)) 表示直线 AB、直线 AC、直线 BC 中的任意一条.

对于某些陈述型几何表述, 例如定理的证明、问题的解答等, 由于现有的几何软件并不能对其进行验证与处理, 因此目前我们所述的几何描述语言并没有包含对它们的形式表示.

4.3.3　其它表述的形式语言

过程化知识表述

定义 4.3.9. *翻译规则*表示将形式化的几何概念实例转化为自然语言表述的规则, 其形式表示为 Translation(C, S, L), 其中 C 是一个概念, S 是一个包含 C 中变量的自然语言语句, L 表示自然语言的类型, 如 english、中文等.

例如,

- Translation(collinear(A::Point, B::Point, C::Point), '$A, B,$ and C are collinear', english) 表示将 collinear(D, E, F) 转化为"$D,\ E,$ and F are collinear";
- Translation(perpendicularline(A::Point, l::Line), '过 A 垂直于 l 的直线', 中文) 表示将 perpendicularline(B, m) 转化为"过 B 垂直于 m 的直线".

定义 4.3.10. *代数化规则*表示将形式化的几何概念实例转化为在笛卡尔坐标系下关于坐标分量以及其它几何量的代数表达式的规则, 其形式如下:

(1) P[n] 表示 P 所对应坐标的第 n 个分量, 其中 P 是概念中被赋予类型 Point 的引用, n 是一个自然数;

(2) AlgebraicScript(C, μ, T) 是一个*代数化规则*, 其中 C 是一个布尔概念, μ 是一个关于 C 中被赋予类型 Point 的引用的坐标分量以及其它几何量的代数表达式, T 表示代数化的类型, 如 Coordinate (坐标代数化), Area (面积量代数化) 等;

(3) AlgebraicScript(C, δ, μ, T) 是一个*代数化规则*, 其中 C 是一个实体概念, δ 是一个坐标, μ 是一个关于 C 中被赋予类型 Point 的引用的坐标分量, δ 的坐标分量以及其它几何量的代数表达式, T 表示代数化的类型.

例如,

- AlgebraicScript(collinear(A::Point, B::Point, C::Point), $-B$[2]*C[1]+B[2]*A[1]+A[2]*C[1]+B[1]*C[2]$-B$[1]*A[2]$-A$[1]*C[2]$=0$, Coordinate) 表示将共线的概念坐标代数化;
- AlgebraicScript(line(A::Point, B::Point), (x, y), x*B[2]$-A$[1]*B[2]$-x$*B[1]+A[1]*B[1]$-A$[2]*$y-A$[1]*B[1]+A[2]*B[1]+A[1]*$y=0$, Coordinate) 表示将直线的概念坐标代数化.

定义 4.3.11. *图形化规则*表示将形式化的概念实例转化为作图指令的规则, 其形式表示为 DiagramScript$(C, \eta, \overline{N})$, 其中 C 是一个实体概念, η 为相应的作图指令, $\overline{N}$ 表示 η 可以被识别的动态几何软件的名字.

例如,

- DiagramScript(perpendicularline(A::Point, l::Line), Perpendicular(l; A), Cinderella);
- DiagramScript(perpendicularline(A::Point, l::Line), PerpendicularLine[A, l], GeoGebra).

代数表述

*代数表述*是指几何命题或者问题在引入的坐标系中所对应的代数形式 (即代数化) 的表述. 目前, OpenMath 在许多项目中被用作数学对象表达的标准语言, 并可以通过 OpenMath Phrasebooks [105] 与计算机代数系统交互, 因此我们选用 OpenMath 来表示关于坐标以及其它几何量的代数表述.

作图表述

定义 4.3.12. *作图表述*是指几何构型的作图步骤, 其形式表示为 Diagram$(N, \{\eta_1; \cdots; \eta_n\}, \overline{N})$, 其中 N 表示此作图表述所作图形的名字, η_i $(1 \leqq i \leqq n)$ 都是作图指令, $\overline{N}$ 表示这些作图指令可以被识别的动态几何软件的名字[注1].

例如, Diagram(Simson's theorem, $\{A = (-11.1, 1.62); B = (-22.84, -15.37); C = (7.14, -15.75)$; t = Polygon[A, B, C]; o = Circle[A, B, C]; D = Point[o]; l = PerpendicularLine[D, Line[A, B]]; a = Angle[Line[A, B], l]; m = PerpendicularLine[D, Line[B, C]]; n = PerpendicularLine[D, Line[A, C]]; J = Intersect[l, Line[A, B]]; H = Intersect[m, Line[B, C]]; G = Intersect[n, Line[A, C]]; $SimsonLine$ = Line[H, G]; }, GeoGebra).

4.3.4 几何知识的自然语言表述

几何知识需要使用自然语言进行表述以方便人们的学习和理解. 为了给人以更好

[注1] 动态几何软件 GeoGebra 的作图指令列表请参见 http://www.geogebra.org/en/wiki/index.php/Command_Descriptions.

的阅读体验, 我们采用 XML 对自然语言表述的文本进行样式化. 例如, 对几何表述中涉及到的数学对象使用 OpenMath 语言或者 LaTeX 语言来表示; 链接使用标签“link”来标注; 数学表达式使用标签“equation”来标注; 静态图片使用标签“figure”来标注; 动态的图形文件 (可以被相应的几何软件载入运行) 使用标签“dynamicFigure”来标注; 段落使用标签“para”来标注等. 关于这些标签的具体使用说明, 请参见附录 **C.**.

4.4　功能设计

下面讨论电子几何教科书系统功能的设计与实现方法. 由于几何知识库已经实现了对知识数据的操作与呈现等管理功能, 因此这里我们主要讨论以下两方面内容: 一方面是研究如何将几何构型、命题以及问题的 GDL 表述自动地同义转化为满足外部几何软件应用和处理需要的表述; 另一方面是研究实时地检测所构建教科书的结构一致性、完备性与冗余性的方法.

4.4.1　实例与概念的匹配

应用几何描述语言, 可以部分地形式化几何知识. 然而, GDL 表述除了应该满足语法上的要求之外, 还需要满足语义上的约束, 即实例的参数需要符合相应的概念对参数*类型*的要求. 例如, 若已定义一个几何概念 triangle(A::Point, B::Point, C::Point), 那么在实例 triangle(a, b, c) 中, a、b、c 需要是类型为 Point 的概念实例才使得此实例有意义. 如果一个概念的定义可以赋予其实例以语义, 我们称这个实例为这个概念的*可解释实例*, 也称这个实例*应用*了这个概念, 或者这个实例与这个概念*匹配*. 另一方面, 概念的实例需要通过它所应用的概念的定义来获得语义上的解释, 或者在文档中呈现为链接以便于人们获取其定义. 我们称在几何描述语言中由实例本身的形式来获得其所应用的概念的问题为*概念匹配问题*. 例如, 在 GDL 表述中, 一个实例表示为 line(D, E), 为了对其进行语义上的处理, 就需要找到这个实例所应用的概念的定义.

为了精确地描述上述概念匹配问题, 我们分别对概念以及概念的实例附加类型操作 ($\texttt{Type}(I)$ 表示 I 的类型), 并通过比较判断这些类型之间的关系来解决这一问题.

定义 4.4.1. 概念的*类型*定义如下:

(1) 一个抽象实例形如 $V :: T$,

$$\texttt{Type}(V :: T) = T;$$

(2) 一个概念形如 $f(a_1, \cdots, a_n)$ $(0 \leqq n)$,

$$\texttt{Type}(f(a_1, \cdots, a_n)) = f(\texttt{Type}(a_1), \cdots, \texttt{Type}(a_n)),$$

其中 $\texttt{Type}(a_i)$ $(1 \leqq i \leqq n)$ 称为其*参数*.

类型含有"( )"的对数叫做类型的*层数*. 由于概念只有有限层数和有限个参数, 因此其类型也只有有限层数和有限个参数.

在几何描述语言中, 实例中的变量或者是某个抽象实例的引用, 或者是引用分句中某个实例的引用, 因此

定义 4.4.2. 实例的*类型*定义如下:

(1) 若 V 是一个变量, 并且存在引用分句 $V := I$ (其中 I 是一个实例), 则

$$\texttt{Type}(V) = \texttt{Type}(I);$$

(2) 若 V 是一个变量, 并且存在抽象实例 $V :: T$, 则

$$\texttt{Type}(V) = T;$$

(3) 若 V 是一个变量, 但不满足 (1) 和 (2) 的情形, 则 $\texttt{Type}(V) = \text{null}$;

(4) $\texttt{Type}(n) = \text{AlgebraicQuantity}$, 其中 n 为一个数量, AlgebraicQuantity 是一个抽象类型;

(5) $\texttt{Type}(\pi) = \text{Degree}$, 其中 Degree 是一个抽象类型;

(6) 若一个实例形如 $f(a_1, ..., a_n)$ $(0 \leqq n)$, 则

$$\texttt{Type}(f(a_1, \cdots, a_n)) = f(\texttt{Type}(a_1), \cdots, \texttt{Type}(a_n)),$$

其中 $\texttt{Type}(a_i)$ $(1 \leqq i \leqq n)$ 称为其*参数*.

类型含有"( )"的对数叫做类型的*层数*. 由于实例只有有限层数和有限个参数, 因此其类型也只有有限层数和有限个参数.

定义 4.4.3. 给定两个类型 T_1 和 T_2 分别形如 $f(a_1, \cdots, a_n)$ 和 $g(b_1, \cdots, b_m)$, 如果满足 f 与 g 相同, $m = n$, $a_i = b_i$ $(1 \leqq i \leqq n)$, 那么我们称 T_1 与 T_2 两个类型*相等*, 记为 $T_1 = T_2$.

例如, 概念 distance(A::Point, B::Point) 的类型为 distance(Point, Point), 概念 distance (A::Point, l::Line) 的类型为 distance(Point, Line). 若一个构型为 configuration(A := point(), B := point(), l := line(A, B), collinear(A, B, C)), 则 A 和 B 的类型都为 point(), l 和 line(A, B) 的类型为 line(point(), point()), 而 collinear(A, B, C) 的类型为 collinear (point(), point(), null). distance(Point, Point) = distance(Point, Point).

定义 4.4.4. 给定概念 C 及实例 I, 如果 $\texttt{Type}(C) = \texttt{Type}(I)$, 那么 I 是 C 的可解释实例.

然而, 由于实例允许嵌套, 定义 4.4.4 并不能涵盖实例与概念相匹配的所有情形. 例如, I 是实例 intersection(line(A, B), line(C, D)), 其中 $\texttt{Type}(A) = \texttt{Type}(B) = \texttt{Type}(C) = \texttt{Type}(D)$ = Point, K 是概念 intersection(l::Line, m::Line), 表示任意两条直线的交点. $\texttt{Type}(I)$ = intersection(line(Point, Point), line(Point, Point)), $\texttt{Type}(K)$ = intersection(Line, Line). 尽管 $\texttt{Type}(I) \neq \texttt{Type}(K)$, 但是过两点的直线也是直线, K 的定义可以用来解释 I, I 也应该是 K 的可解释实例. 因此仅仅通过判断类型是否相等并不足以判断实例与概念是否匹配. 我们通过引入类型之间的序来解决这个问题.

定义 4.4.5. 给定一个脚本 $\mathbb{S}$, 类型的序 ($\prec$) 定义如下:

(1) 若 T_1 和 T_2 均为抽象类型, 它们所对应的本原或抽象概念分别为 C_1 和 C_2, 并且在第 3.2.3 节继承关系图 (图 3.3) 中 C_1 和 C_2 的定义构成继承关系, 即 $C_1 \to C_2$ 存在, 则 $T_1 \prec T_2$;

(2) 若 $\mathbb{D}$ 是 $\mathbb{S}$ 中的一个定义, C 为 $\mathbb{D}$ 的目标概念, $\mathbb{R}$ 为 $\mathbb{D}$ 的返回体, I 为 $\mathbb{R}$ 的中心实例 (此时 $\mathbb{R}$ 为约束绑定或逆绑定),

- 若 $\mathbb{R}$ 为抽象实例 (此时 I 不存在), 则 $\texttt{Type}(C) \prec \texttt{Type}(\mathbb{R})$;
- 若 I 是形如 $l(\alpha_1, \cdots, \alpha_n)$ 的复合布尔分句, 则 $\texttt{Type}(C) \prec \texttt{Type}(\alpha_i)$ [注 1] $(1 \leq i \leq n)$;
- 若 I 为一个变量, 或者抽象实例, 或者某个概念的实例, 或者非复合布尔分句, 则 $\texttt{Type}(C) \prec \texttt{Type}(I)$;
- 若 $\mathbb{R}$ 是一个集合返回体 (此时 I 不存在), 则对 $\mathbb{R}$ 的每个分量, 依据上述的定义便可得到相应类型的序;

  这些序都称为由 $\mathbb{D}$ 导出的类型序;

[注 1] 若这里 α_i 仍是复合布尔分句, 则对 α_i 执行此步骤.

(3) 给定一个类型形如 $f(a_1, \cdots, a_n)$ $(1 \leqq n)$, 若存在类型 T 及参数 a_i $(1 \leqq i \leqq n)$ 满足 $\texttt{Type}(a_i) \prec T$, 那么 $f(a_1, ..., a_n) \prec f(a_1, \cdots, T, \cdots, a_n)$, 其中 T 是第 i 个参数;

(4) 如果 T_1、T_2、T_3 都是类型, $T_1 \prec T_2$ 且 $T_2 \prec T_3$, 则 $T_1 \prec T_3$;

$T_1 \prec T_2$ 被称为 T_2 *提升* T_1, T_2 称为 T_1 的*父类型*, 记为 $T_2 = \texttt{Father}(T_1)$. 一个类型可以有多个父类型.

例如, line(Point, Point) $\prec$ Line 以及 intersection(line(Point, Point), line(Point, Point)) $\prec$ intersection(Line, line(Point, Point)) $\prec$ intersection(Line, Line).

断言 4.4.1. 抽象类型个数有限, 并且对任何一个抽象类型 T_A, 都有 $\texttt{Father}(T_A) \neq T_A$.

定理 4.4.1. 给定一个脚本 $\mathbb{S}$, 存在一个实例 I 满足 $\texttt{Father}(\texttt{Type}(I)) = \texttt{Type}(I)$ 当且仅当存在 $\mathbb{S}$ 中的定义 $\mathbb{D}$ 满足 $\texttt{Father}(\texttt{Type}(\texttt{TC}(\mathbb{D}))) = \texttt{Type}(\texttt{TC}(\mathbb{D}))$.

证明: 充分性:

由定义 4.4.2 可知 $\texttt{Type}(I)$ 一定形如 $f(a_1, \cdots, a_n)$ $(0 \leqq n)$. 由于 $\texttt{Father}(\texttt{Type}(I)) = \texttt{Type}(I)$, 若存在定义 $\mathbb{D}$ 满足 $\texttt{Type}(I) = \texttt{Type}(\texttt{TC}(\mathbb{D}))$, 则结论显然成立; 否则, $1 \leqq n$ 一定成立并且一定存在 I 的某个参数 a_i $(1 \leqq i \leqq n)$ 和类型 T', 满足 $a_i \prec T'$ 且 $f(a_1, \cdots, a_i, \cdots, a_n) \prec f(a_1, \cdots, T', \cdots, a_n) \prec f(a_1, \cdots, a_i, \cdots, a_n)$.

(1) 若存在定义 $\mathbb{D}$ 满足 $f(a_1, \cdots, T', \cdots, a_n) = \texttt{Type}(\texttt{TC}(\mathbb{D}))$, 则结论成立;

(2) 否则, $T' \prec a_i$ 并且 $\texttt{Father}(a_i) = a_i$. 对 a_i 重复以上过程, 而每重复一次, $\texttt{L}(a_i)$ 都比 $\texttt{L}(I)$ 少 1. 因为实例的层数是有限的, 假设以上过程可以被重复直到层数为 1, 即存在一个形如 $g(b_1, \cdots, b_m)$ $(1 \leqq m)$ 的参数满足对任意定义 $\mathbb{D}$ 都有 $g(b_1, \cdots, b_m) \neq \texttt{Type}(\texttt{TC}(\mathbb{D}))$, 其中 $\texttt{L}(b_k) = 0$ (即 b_k 是一个抽象类型, $1 \leqq k \leqq m$). 一方面, 从前面的讨论可知, 一定存在某个参数 b_j 满足 $\texttt{Father}(b_j) = b_j$; 另一方面, 由断言 4.4.1 可知, 对所有 k 都有 $\texttt{Father}(b_k) \neq b_k$. 这样便产生了矛盾. 因此, 在层数减少到 0 之前, 一定存在某个参数 a' 和某个定义 $\mathbb{D}'$ 满足 $a' = \texttt{Type}(\texttt{TC}(\mathbb{D}'))$ 并且 $\texttt{Father}(\texttt{Type}(\texttt{TC}(\mathbb{D}'))) = \texttt{Type}(\texttt{TC}(\mathbb{D}'))$.

必要性:

由定义 4.4.5 可知存在某个类型 T' 满足 $\texttt{Type}(\texttt{TC}(\mathbb{D})) \prec T' \prec \texttt{Type}(\texttt{TC}(\mathbb{D}))$. 一定存在某个实例 I, $\texttt{Type}(I)$ 形如 $g(b_1, \cdots, \texttt{Type}(\texttt{TC}(\mathbb{D})), \cdots, b_m)$ 并且满足 $\texttt{Type}(I) \prec g(b_1, \cdots, T', \cdots, b_m) \prec g(b_1, \cdots, \texttt{Type}(\texttt{TC}(\mathbb{D})), \cdots, b_m) = \texttt{Type}(I)$. 因此结论成立. □

推论 4.4.1. 对一个脚本 $\mathbb{S}$ 中的任意定义 $\mathbb{D}$, 如果 $\texttt{Father}(\texttt{Type}(\texttt{TC}(\mathbb{D}))) \neq \texttt{Type}(\texttt{TC}(\mathbb{D}))$, 那么对任一实例 I, 若 $\texttt{Father}(\texttt{Type}(I))$ 存在, 则 $\texttt{Father}(\texttt{Type}(I)) \neq \texttt{Type}(I)$ 并且 $\texttt{Father}(\texttt{Type}(I))$ 的个数有限. 我们称这样的脚本是*良好的* (*well-formed*).

证明: 由定理 4.4.1 可知, 对任一实例 I, 若 $\texttt{Father}(\texttt{Type}(I))$ 存在, 则 $\texttt{Father}(\texttt{Type}(I)) \neq \texttt{Type}(I)$. $\texttt{Father}(\texttt{Type}(I))$ 只可能存在如下两种情形:

- 若存在定义 $\mathbb{D}$ 满足 $\texttt{Father}(\texttt{Type}(I)) = \texttt{Type}(\texttt{TC}(\mathbb{D}))$, 那么 $\texttt{Father}(\texttt{Type}(I))$ 的个数是有限的, 这是因为 $\texttt{Father}(\texttt{Type}(\texttt{TC}(\mathbb{D}))) \neq \texttt{Type}(\texttt{TC}(\mathbb{D}))$ 并且脚本中定义的个数是有限的;
- 否则, $\texttt{Type}(I)$ 的某个参数存在父类型并且这个参数的层数比 $\texttt{Type}(I)$ 的层数少 1. 由于 $\texttt{Type}(I)$ 的参数个数有限并且层数也有限, 因此由断言 4.4.1 可知, 这种情形下 $\texttt{Father}(\texttt{Type}(I))$ 的个数也是有限的.

综上, 对任一实例 I, 若 $\texttt{Father}(\texttt{Type}(I))$ 存在, 则其个数有限. □

事实上, 脚本是否良好反应了概念是否被循环定义. 因此检测一个脚本是否是良好的对于知识管理有着重要的意义. 由于脚本只包含有限个定义, 因此可以遍历这些定义的目标概念的类型, 依次判断是否存在某个父类型与其相等. 这种判断通常比较简单, 可以人为进行.

定义 4.4.6. 给定脚本 $\mathbb{S}$、概念 C 及实例 I, 如果 $\texttt{Type}(I) \preceq \texttt{Type}(C)$, 并且 I 与 C 使用的词汇相同, 那么 I 称为 C 的*可解释实例*, 或者称 I *应用* C, I 与 C 相*匹配*.

注 4.4.1. 由于实例的类型的父类型有多个, 因此实例所应用的概念可以不唯一.

下面, 我们将概念匹配问题精确地表述如下:

问题 4.4.1 (概念匹配问题). 给定一个实例 I 和一个良好的脚本 $\text{Script}(\mathbb{D}_1, \cdots, \mathbb{D}_n)$, 找到所有的 $\mathbb{D}_i$ $(1 \leqq i \leqq n)$ 使得 I 与 $\mathbb{D}_i$ 的目标概念相匹配.

根据定义 4.4.6, 只要一个定义的目标概念的类型为一个实例的类型的父类型, 那么这个实例便应用这个目标概念. 因此, 对于给定的实例, 需要遍历其类型的父类型, 然后检测每个父类型是否与某个定义的目标概念的类型相等进而找到这个实例所应用的概念 (集合). 由推论 4.4.1 可知, 如果脚本是良好的, 那么实例的类型的父类型只有有限个. 因此我们首先给出父类型遍历问题以及父类型的遍历算法.

问题 4.4.2 (**父类型遍历问题**). 给定一个实例 I 和一个良好的脚本 $\mathbb{S}$, 遍历 I 的类型的父类型.

算法 1: **父类型遍历算法 (FTT)**

输入: 实例 I, 良好的脚本 $\mathbb{S}$
输出: $\texttt{Father}(\texttt{Type}(I))$ 的集合

```
result := null;
FIs := null;
T := Type(I);
if T 存在 then
    if I 形如 f(a_1, ··· , a_n) then
        if n ≥ 1 then
            for i ← 1 to n do
                F_i := FTT(a_i, S);
                T_i := Type(a_i);
            L := (T_1 ∪ F_1) × ··· × (T_n ∪ F_n);          // × 表示集合的笛卡尔乘积
            for l in L do
                T' := f + l ;                               // + 表示将 f 与 l 相连接
                if T ≠ T' then
                    将 T' 不重复地添加到 result 中;
                for D in S do
                    if T' = Type(TC(D)) then
                        RS := null;
                        if RB(D) 是一个集合返回体 then
                            将 RB(D) 的所有分量不重复地添加到 RS 中;
                        else
                            将 RB(D) 不重复地添加到 RS 中;
                        for R in RS do
                            if R 是一个抽象实例 then
                                将 R 不重复地添加到 FIs 中;
                            else
                                RI := CI(R);
                                if RI 是一个复合布尔分句 then
                                    将 RI 的所有实例参数不重复地添加到 FIs 中;
                                else
                                    将 RI 不重复地添加到 FIs 中;
                            for E in FIs do
                                将 Type(E) 不重复地添加到 result 中;
                                将 FTT(E,S) 的全部元素不重复地添加到 result 中;
                        break;
        else
            for D in S do
                if T = Type(TC(D)) then
                    将 Type(RB(D)) 不重复地添加到 result 中;     // RB(D) 是抽象实例
                    将 FTT(RB(D), S) 的全部元素不重复地添加到 result 中;
                    break;
    else
        将所有 Father(T) 不重复地添加到 result 中;          // 这里 T 是一个抽象类型
返回 result;
```

注 4.4.2. 在算法 1 中, 根据定义 4.4.5 的第 (3) 条可知, 第 14 行中 T' 是 T 的父类型; 根据第 (2) 条可知, 第 32 行中 $\texttt{Type}(E)$ 和第 38 行中 $\texttt{Type}(\texttt{RB}(\mathbb{D}))$ 也是 T 的父类型; 根据第 (4) 条可知, 第 33 行和 39 行中递归调用的所有结果都是 T 的父类型; 第 42 行中抽象类型 T 的所有父类型都被添加到 $result$ 中. 因此, $result$ 中的所有元素都是 $\texttt{Type}(I)$ 的父类型. 另一方面, 根据定义 4.4.5 可知, $\texttt{Type}(I)$ 的所有父类型都被考虑过了. 因此, 算法 1 的输出是 $\texttt{Type}(I)$ 的所有父类型的集合.

由于良好的脚本 $\mathbb{S}$ 所包含定义的个数, 实例的参数个数, 以及实例的类型的父类型个数都是有限的 (由推论 4.4.1 可知), 因此父类型遍历算法 1 经过有限步之后将终止.

应用父类型遍历算法, 概念匹配问题可以得到解决, 算法如 2.

算法 2: 概念匹配算法 (CMA)

输入: 实例 I, 良好的脚本 $\mathbb{S}$
输出: $\mathbb{S}$ 中所有与 I 相匹配的概念的定义的集合

```
1  result := null;
2  T := Type(I);
3  if T 存在 then
4      for 𝔻 in 𝕊 do
5          if T = Type(TC(𝔻)) then
6              将 𝔻 不重复地添加到 result 中;
7      if I 形如 f(a_1, ··· , a_n) then
8          if n ≥ 1 then
9              for i ← 1 to n do
10                 F_i := FTT(a_i, 𝕊);
11                 T_i := Type(a_i);
12             L := (T_1 ∪ F_1) × ··· × (T_n ∪ F_n) ;        // × 表示集合的笛卡尔乘积
13             for l in L do
14                 T' := f + l ;                              // + 表示将 f 与 l 相连接
15                 if T ≠ T' then
16                     for 𝔻 in 𝕊 do
17                         if T' = Type(TC(𝔻)) then
18                             将 𝔻 不重复地添加到 result 中;
19                             break;
20 返回 result;
```

注 4.4.3. 算法 2 的正确性是显然的. 与父类型遍历算法类似, 由于良好的脚本所包含定义的个数, 实例的参数个数, 以及实例的类型的父类型个数都是有限的, 因此概念匹配算法经过有限步之后将终止. 应用概念匹配算法, 可以检测 GDL 表述中实例的使用是否正确. 如果一个实例应用了脚本中的某个概念, 那么它的使用便是正确的; 否则, 需要修改此实例或者向脚本中增加与此实例相匹配的概念的定义来进行修正.

4.4.2 几何表述的同义转化

为了使表述简洁清晰, 需要尽可能地应用导出 (复杂) 概念. 然而在获得一个表述的语义时就需要使用概念的定义将导出概念替换为简单概念, 这也是我们在进行计算、证明等过程中经常使用的手段. 几何描述语言的一个重要特征就是将实例与其应用的概念定义在同一框架下表述, 从而使得所表述的对象可以获得语义上的解释. 因此, 对于几何描述语言, 需要研究如何根据概念的定义将导出概念的实例同义转化为简单概念的实例, 以使转化所得表述可以被有效地使用. 例如, 根据定义 $\mathbb{D}_1$、$\mathbb{D}_2$、$\mathbb{D}_3$、$\mathbb{D}_4$ 可以对实例 foot(D, line(E, F)) 进行如下转化: foot(D, line(E, F)) $\xrightarrow{\mathbb{D}_1}$ foot(D, line(E, F)) $\xrightarrow{\mathbb{D}_2}$ [intersection(perpendicularline(D, line(E, F)), line(E, F))] $\xrightarrow{\mathbb{D}_3, \mathbb{D}_4}$ [v_1::Point where and(incident(D, v_0), perpendicular(v_0, line(E, F)), incident(v_1, v_0), incident(v_1, line(E, F)))]. 其中 v_0 和 v_1 是新引入的变量, 并且

$\mathbb{D}_1$: Definition(line(A::Point, B::Point), l::Line, null);

$\mathbb{D}_2$: Definition(foot(A::Point, l::Line), [intersection(perpendicularline(A, l), l)], null);

$\mathbb{D}_3$: Definition(perpendicularline(A::Point, l::Line), [m::Line where and(incident(A, m), perpendicular(m, l))], null);

$\mathbb{D}_4$: Definition(intersection(l::Line, m::Line), [A::Point where and(incident(A, l), incident(A, m))], null).

以上转化所得表述中的实例都是简单概念的实例, 并且其中的约束关系 (和构造方式) 都得到保留, 可以更容易地被操作处理和使用, 如与外部几何软件包进行交互.

为了精确地描述并实现 GDL 表述的这种转化, 我们在几何描述语言的语法基础上引入如下概念来形式化这一过程, 进而提出同义转化的算法. 由于概念的实例是几何描述语言的主要成分, 因此我们先讨论概念实例的转化, 接下来再讨论 GDL 表述中其它结构的转化.

定义 4.4.7. *替换*是将一个 GDL 表述中的变量用其它项或者实例代替而得到一个新表述的操作. 设 F 是一个 GDL 表述, a 是其中的变量, b 是一个项或者实例, 则用 b 代替 a 的替换操作可表示为

$$F\langle a \to b\rangle = F(b).$$

例如, 点 A 到直线 l 的垂足可用实例 intersection(perpendicularline$(A,l),l$) 来表述 (意思是过 A 与 l 垂直的直线与 l 的交点). 假设变量 A 需要被另一个变量 D 代替, 变量 l 需要用实例 line(E,F) 来代替, 则此替换操作可表示为

$$\text{intersection}(\text{perpendicularline}(A,l),l)\langle A\to D, l\to \text{line}(E,F)\rangle =$$
$$\text{intersection}(\text{perpendicularline}(D,\text{line}(E,F)),\text{line}(E,F)).$$

替换操作并不能随意地进行, 而需要按照一定的规则, 以保证替换后的结果与原表述同义并满足应用的要求. 我们引入下面重要的概念, 即实例的转化与化简来形式地描述这一过程.

定义 4.4.8. 实例的*转化*是指按照规则将实例转化为具有特定形式表述的操作. 一个实例 I 称为在脚本 $\mathbb{S}$ 下按照转化规则 $C\mapsto R$ 是*可被转化的*, 若 $\mathbb{S}$ 中存在该实例所应用的概念 C, 并且如下条件满足: 设 $A_1,\cdots,A_n$ $(1\leq n)$ 是 C 中的抽象实例, $V_1,\cdots,V_n$ 分别是 $A_1,\cdots,A_n$ 的引用, 存在项或者实例 $B_1,\cdots,B_n$ 满足 $C\langle V_1\to B_1,\cdots,V_n\to B_n\rangle = I$. 那么 I 的转化可表示为

$$I\mapsto R\langle V_1\to B_1,\cdots,V_n\to B_n\rangle,$$

其中 R 称为转化的*模板*, $R\langle V_1\to B_1,\cdots,V_n\to B_n\rangle$ 记为 $\texttt{Transform}(I,C,R)$.

定义 4.4.9. 实例的*化简*是根据实例所应用的概念的定义, 将实例转化为具有定义返回体形式的一种特殊的实例转化操作. 一个实例 I 称为在脚本 $\mathbb{S}$ 下根据 Definition$(C,\mathbb{R},\mathbb{N})$ 是*可被化简的*, 若 $\mathbb{S}$ 是良好的, 并且如下条件满足:

- I 在 $\mathbb{S}$ 下按照 $C\mapsto\mathbb{R}$ 是可被转化的;
- C 不是基本概念;
- Definition$(C,\mathbb{R},\mathbb{N})$ 存在于 $\mathbb{S}$ 中.

实例 I 的化简可表示为

$$I\mapsto\mathbb{R}\langle V_1\to B_1,\cdots,V_n\to B_n\rangle\langle p_0\to v_0,\cdots,p_m\to v_m\rangle,$$

其中 $p_0,\cdots,p_m$ 是 $\mathbb{R}$ 中出现的抽象实例的引用, $v_0,\cdots,v_m$ 为新引入的变量, 即与 I 中的变量都不相同.

例如, 实例 foot(D, line(E, F)) 根据前述的定义 $\mathbb{D}_2$ 有如下化简:

$$\text{foot}(D, \text{line}(E, F)) \mapsto [\text{intersection}(\text{perpendicularline}(D, \text{line}(E, F)), \text{line}(E, F))].$$

同样根据前述的定义 $\mathbb{D}_4$ 有如下化简:

$$\text{intersection}(\text{perpendicularline}(D, \text{line}(E, F)), \text{line}(E, F)) \mapsto [A::\text{Point where}$$
$$\text{and}(\text{incident}(A, \text{perpendicularline}(D, \text{line}(E, F))), \text{incident}(A, \text{line}(E, F)))]\langle A \to new_0 \rangle$$
$$= [new_0::\text{Point where and}(\text{incident}(new_0, \text{perpendicularline}(D, \text{line}(E, F))),$$
$$\text{incident}(new_0, \text{line}(E, F)))].$$

实例的化简操作实质上反应了根据它所应用的概念定义来对实例进行语义解释的过程, 保证了实例所具有的约束关系 (和构造方式) 不变, 因此化简得到的表述与原实例所表达的语义是一致的或者说是同义的, 并且形式已不再是实例的形式, 而与模板的形式相同. 一个实例经过化简之后得到的表述会包含新的实例, 需要被继续化简, 因此对实例通常可以进行一系列化简. 下面, 我们将"化简"的概念推广到整个几何描述语言.

定义 4.4.10. 一个实例、分句、构型、命题、问题以及其他 GDL 表述 $\mathcal{S}$ 称为在脚本 $\mathbb{S}$ 下是*可规约的*, 若至少存在一个 $\mathcal{S}$ 中所出现的实例 (或子实例) 在 $\mathbb{S}$ 下根据某个定义可被化简. 对这个实例的化简也称为对 $\mathcal{S}$ 的化简. 如果对 $\mathcal{S}$ 进行一系列化简 (称为*规约*) 直到 $\mathcal{S}'$ 不可被化简, 那么这一过程称为将 $\mathcal{S}$ *规约为* $\mathcal{S}'$, $\mathcal{S}'$ 称为 $\mathcal{S}$ 在 $\mathbb{S}$ 下的*规约结果*.

为了使用同一个程序对各种不同形式的表述进行规约处理, 我们引入一个规范的形式, 即每一步化简操作的结果都要转化为这种形式, 以便进行下一步化简.

定义 4.4.11. 形如 $[\mathcal{A}, \mathcal{C}, \mathcal{P}, \mathcal{N}]$ 的表述 $\mathcal{S}$ 称为*正规型*, 其中 $\mathcal{A}$ 是一个变量, 实例, 分句, 或者构型, 称为*主体*, 记为 $\texttt{getI}(\mathcal{S})$; $\mathcal{C}$ 是一个布尔分句, 引用分句, 或者 null, 表示对 $\mathcal{A}$ 中变量的约束 (或构造), 称为*约束*, 记为 $\texttt{getC}(\mathcal{S})$; $\mathcal{P}$ 是一个构型或者 null, 表示 $\mathcal{A}$ 中某些变量的构造方式, 称为*语境*, 记为 $\texttt{getP}(\mathcal{S})$; $\mathcal{N}$ 是一个布尔分句或者 null, 表示 $\mathcal{A}$ 中变量有意义所应满足的条件, 称为*非退化条件*, 记为 $\texttt{getN}(\mathcal{S})$. $\mathcal{A}$、$\mathcal{C}$、$\mathcal{P}$、$\mathcal{N}$ 称为正规型的分量.

由于几何描述语言允许集合表示, 因此我们将正规型作如下扩展:

(1) 若 $\alpha_1, \cdots, \alpha_n$ 都为正规型, 那么形如 $\{\alpha_1; \cdots; \alpha_n\}$ 的 $\mathcal{S}$ 是一个 (集合) *正规型*, 这时 $\texttt{getI}(\mathcal{S}) = \mathcal{S}$, $\texttt{getC}(\mathcal{S}) = \texttt{getP}(\mathcal{S}) = \texttt{getN}(\mathcal{S}) =$null. 若 $\alpha_1, \cdots, \alpha_n$ 中所有不为 null 的约束可以同时成立, 则这样的集合正规型称为*共存正规型*, 否则称为*互斥正规型*;

(2) 若 $[\mathcal{A}, \mathcal{C}, \mathcal{P}, \mathcal{N}]$ 为正规型, 那么 $\mathcal{A}$、$\mathcal{C}$、$\mathcal{P}$、$\mathcal{N}$ 中的某些变量或全部变量替换为集合正规型所得表述也是*正规型*;

(3) 若 $\mathcal{S}$ 不是正规型, 则 $\texttt{getI}(\mathcal{S}) = \mathcal{S}$, $\texttt{getC}(\mathcal{S}) = \texttt{getP}(\mathcal{S}) = \texttt{getN}(\mathcal{S}) = \text{null}$.

给定一个正规型 $\mathcal{S}$, $\texttt{getI}(\mathcal{S})$ 中含有 { } 的对数称为集合层数, 记为 $\texttt{SL}(\mathcal{S})$. 集合层数是有限的.

定义 4.4.12. 表述 $\mathcal{S}$ 的*正规化*是指将 $\mathcal{S}$ 转化为正规型的一系列操作. 我们通过全面分析在规约过程中可能产生的不同形式表述, 引入如下正规化规则, 其中 $[\mathcal{A}, \mathcal{C}, \mathcal{P}, \mathcal{N}]$ 和 $[\mathcal{A}', \mathcal{C}', \mathcal{P}', \mathcal{N}']$ 都是正规型.

(1) 若 $\mathcal{S}$ 形如 $[V]$, V 是一个变量, 则正规化可表示为

$$[V] \mapsto [V, \text{null}, \text{null}, \text{null}];$$

(2) 若 $\mathcal{S}$ 形如 $[M \text{ where } Q]$, 且由实例 I 化简而得, 若其中 M 是一个变量、实例或者布尔分句, 则正规化可表示为 $[M \text{ where } Q] \mapsto [M, Q, \text{null}, \text{null}]$; 若其中 M 是一个构型, 则正规化可表示为 $[M \text{ where } Q] \mapsto [I, Q, M, \text{null}]$;

(3) 若 $\mathcal{S}$ 形如 $[[\mathcal{A}', \mathcal{C}', \mathcal{P}', \mathcal{N}'], \mathcal{C}, \mathcal{P}, \mathcal{N}]$, 则正规化可表示为

$$[[\mathcal{A}', \mathcal{C}', \mathcal{P}', \mathcal{N}'], \mathcal{C}, \mathcal{P}, \mathcal{N}] \mapsto [\mathcal{A}', \text{and}(\mathcal{C}', \mathcal{C}), \mathcal{P} + \mathcal{P}', \text{and}(\mathcal{N}, \mathcal{N}')];$$

(4) 若 $\mathcal{S}$ 形如 $[\mathcal{A}, [\mathcal{A}', \mathcal{C}', \mathcal{P}', \mathcal{N}'], \mathcal{P}, \mathcal{N}]$, 则正规化可表示为

$$[\mathcal{A}, [\mathcal{A}', \mathcal{C}', \mathcal{P}', \mathcal{N}'], \mathcal{P}, \mathcal{N}] \mapsto [\mathcal{A}, \text{and}(\mathcal{A}', \mathcal{C}'), \mathcal{P} + \mathcal{P}', \text{and}(\mathcal{N}, \mathcal{N}')];$$

(5) 若 $\mathcal{S}$ 形如 $[\mathcal{A}, \mathcal{C}, [\mathcal{A}', \mathcal{C}', \mathcal{P}', \mathcal{N}'], \mathcal{N}]$, 则正规化可表示为

$$[\mathcal{A}, \mathcal{C}, [\mathcal{A}', \mathcal{C}', \mathcal{P}', \mathcal{N}'], \mathcal{N}] \mapsto [\mathcal{A}, \text{and}(\mathcal{C}, \mathcal{C}'), \mathcal{P}' + \mathcal{A}', \text{and}(\mathcal{N}, \mathcal{N}')];$$

(6) 若 $\mathcal{S}$ 形如 $[\mathcal{A}, \mathcal{C}, \mathcal{P}, [\mathcal{A}', \mathcal{C}', \mathcal{P}', \mathcal{N}']]$, 则正规化可表示为

$$[\mathcal{A}, \mathcal{C}, \mathcal{P}, [\mathcal{A}', \mathcal{C}', \mathcal{P}', \mathcal{N}']] \mapsto [\mathcal{A}, \text{and}(\mathcal{C}, \mathcal{C}'), \mathcal{P} + \mathcal{P}', \text{and}(\mathcal{A}', \mathcal{N}')];$$

(7) 若 $\mathcal{S}$ 形如 $f(\alpha_1,\cdots,\alpha_n)$, 其中 α_i $(1 \leqq i \leqq n)$ 都是正规型并且不全是集合正规型, 则正规化可表示为

$$f(\alpha_1,\cdots,\alpha_n) \mapsto [f(\texttt{getI}(\alpha_1),\cdots,\texttt{getI}(\alpha_n)), \textsf{and}(\texttt{getC}(\alpha_1),\cdots,\texttt{getC}(\alpha_n)), \texttt{getP}(\alpha_1)+\cdots+\texttt{getP}(\alpha_n), \textsf{and}(\texttt{getN}(\alpha_1),\cdots,\texttt{getN}(\alpha_n))];$$

(8) 若 $\mathcal{S}$ 形如 $V := \mathcal{K}$, 其中 $\mathcal{K}$ 是正规型, V 是变量, 则正规化可表示为

$$V := \mathcal{K} \mapsto [V := \texttt{getI}(\mathcal{K}), \texttt{getC}(\mathcal{K}), \texttt{getP}(\mathcal{K}), \texttt{getN}(\mathcal{K})];$$

(9) 若 $\mathcal{S}$ 形如 $[\mathcal{A},\mathcal{C},\mathcal{P},\mathcal{N}]$, 其中 $\mathcal{A}$ 为布尔分句, 或者其中某些参数或全部参数替换为集合正规型的布尔分句, 则正规化可表示为

$$[\mathcal{A},\mathcal{C},\mathcal{P},\mathcal{N}] \mapsto [\textsf{and}(\mathcal{A},\mathcal{C}), \textsf{null}, \mathcal{P}, \mathcal{N}];$$

(10) 若 $\mathcal{S}$ 形如 $[\mathcal{A},\mathcal{C},\mathcal{P},\mathcal{N}]$, 其中 $\mathcal{A}$ 为构型, 或者是某些参数或全部参数替换为集合正规型的构型, 则正规化可表示为

$$[\mathcal{A},\mathcal{C},\mathcal{P},\mathcal{N}] \mapsto [\mathcal{P}+\mathcal{A}, \mathcal{C}, \textsf{null}, \mathcal{N}];$$

(11) 若 $\mathcal{S}$ 形如 $f(\{\alpha_1;\cdots;\alpha_n\})$, 其中 α_i $(1 \leqq i \leqq n)$ 都是正规型, f 不是 and、or、not、configuration、give 或者 declare, 则正规化可表示为

$$f(\{\alpha_1;\cdots;\alpha_n\}) \mapsto f(\alpha_1,\cdots,\alpha_n);$$

(12) 若 $\mathcal{S}$ 形如 $f(\{\alpha_1;\cdots;\alpha_n\},\cdots,\{\beta_1;\cdots;\beta_n\})$, 其中 α_i $(1 \leqq i \leqq n)$,$\cdots$,β_j $(1 \leqq j \leqq n)$ 都是正规型, f 不是 and、or、not、configuration、give 或者 declare, 则正规化可表示为

$$f(\{\alpha_1;\cdots;\alpha_n\},\cdots,\{\beta_1;\cdots;\beta_n\}) \mapsto \{f(\alpha_1,\cdots,\beta_1);\cdots;f(\alpha_n,\cdots,\beta_n)\};$$

(13) 若 $\mathcal{S}$ 形如 $\{V_1;\cdots;V_n\} := \{\alpha_1;\cdots;\alpha_n\}$, 其中 V_i $(1 \leqq i \leqq n)$ 都是变量, α_j $(1 \leqq j \leqq n)$ 都是正规型, 则正规化可表示为

$$\{V_1;\cdots;V_n\} := \{\alpha_1;\cdots;\alpha_n\} \mapsto \{V_1 := \alpha_1;\cdots;V_n := \alpha_n\}.$$

如果一个表述中的某些部分可以按照以上规则进行正规化操作, 那么称这个表述*可被正规化*. 若 $\mathcal{S}$ 可被正规化, 则其正规化结果记为 $\texttt{Normalize}(\mathcal{S})$. 若 $\mathcal{S}$ 不可被正规化但为正规型, 则 $\texttt{Normalize}(\mathcal{S}) = \mathcal{S}$.

引理 4.4.1. 若 $\mathcal{S}$ 可被正规化, 则正规化操作经过有限步一定终止, 并且正规化结果 $\texttt{Normalize}(\mathcal{S})$ 是正规型.

证明: 我们通过分析定义 4.4.12 中的正规化规则来讨论正规化操作的可终止性. 由于 $[\mathcal{A},\mathcal{C},\mathcal{P},\mathcal{N}]$ 和 $[\mathcal{A}',\mathcal{C}',\mathcal{P}',\mathcal{N}']$ 都是正规型, 若按照规则 (9) 或者 (10) 进行正规化操作, 则所得结果是正规型. 虽然结果按照规则 (9) 或者 (10) 仍然可被正规化, 但是对 null 的操作将终止[注 1], 从而正规化操作也将终止. 若按照规则 (1) 或者 (2) 进行正规化操作, 则所得结果显然是正规型并且可能按照规则 (9) 可被正规化. 由前面的讨论可知, 结论成立. 若按照规则 (3)、(4)、(5) 或者 (6) 进行正规化操作, 则所得结果是正规型并且可能按照规则 (9) 或者 (10) 可被正规化. 由前面的讨论可知, 结论成立. 若按照规则 (7) 进行正规化操作, 则所得结果是正规型. 由于 α_i $(1 \leqq i \leqq n)$ 都是正规型, 因此可能会出现以下三种情形:

- 若所得结果不可被正规化, 则结论成立;
- 若所得结果按照规则 (9) 或者 (10) 可被正规化, 则由前面的讨论可知, 结论成立;
- 若所得结果按照规则 (11) 或者 (12) 可被正规化, 则所得结果可能按照规则 (11)、(12) 或者 (7) 仍然可被正规化, 因此正规化操作需要进一步进行. 按照规则 (11) 或者 (12) 进行正规化操作后, 集合层数将至少减 1. 由于集合层数有限, 按照规则 (11) 或者 (12) 进行有限步正规化操作之后的结果一定不可以再按照规则 (11) 或者 (12) 被正规化. 因此, 按照规则 (7) 进行正规化操作必将终止并且所得结果是正规型.

若按照规则 (8) 进行正规化操作, 则所得结果是正规型并且不可被正规化. 因此, 正规化操作将终止. 若按照规则 (13) 进行正规化操作, 则所得结果可能按照规则 (8) 可被正规化. 综上所述, 结论对所有正规化规则成立. 引理得证. □

事实上, 正如 4.3.1 节所述, 为了应用外部几何软件来处理几何构型、命题或问题的 GDL 表述, 首先要得到其在脚本下的规约结果. 我们应用前述的化简以及正规化操作, 采用内层优先 (深度优先遍历) 的策略来解决规约问题.

[注 1] 例如, $\text{and}(\mathcal{A}, \text{null})$ 和 $\text{and}(\text{null}, \mathcal{A})$ 将返回 $\mathcal{A}$ 并终止当前操作; $\mathcal{P} + \text{null}$ 和 $\text{null} + \mathcal{P}$ 将返回 $\mathcal{P}$ 并终止当前操作.

问题 4.4.3 (**分句规约问题**). 给定一个分句 I (包括集合分句) 和一个良好的脚本 $\mathbb{S}$, 求 I 在 $\mathbb{S}$ 下的规约结果.

算法 3: 分句规约算法 (CRA)

输入: 分句 I, 脚本 $\mathbb{S}$
输出: I 在 $\mathbb{S}$ 下的规约结果

$r :=$ null;
if I 形如 $f(a_1, \cdots, a_n)$ **then**
  if I 形如 choosediff($\{\alpha_1; \cdots; \alpha_n\}, m$) **then**
    $r := \texttt{CRA}(\{\alpha_{i_1}; \cdots; \alpha_{i_m}\})$ $(1 \le i_1, \cdots, i_m \le n$ 且 $i_j \ne i_k)$;
  else
    $r := \texttt{Normalize}(f(\texttt{CRA}(a_1), \cdots, \texttt{CRA}(a_n)))$;
    if r形如$\{\beta_1; \cdots; \beta_k\}$ **then**
      $r := \texttt{CRA}(r)$;
    if $\texttt{SL}(r) = 0$ **then**
      $A := \texttt{getI}(r)$;
      if A 的形式不是 and$(\cdots)$ 或者 or$(\cdots)$ 或者 not$(\cdots)$ 或者 configuration$(\cdots)$ 或者 give$(\cdots)$ 或者 declare$(\cdots)$ **then**
        $defs := \texttt{CMA}(A, \mathbb{S})$;
        for def in $defs$ **do**
          if A 在 $\mathbb{S}$ 下按照 def 是可被化简的 **then**
            $C := \texttt{getC}(r)$; $P := \texttt{getP}(r)$; $N := \texttt{getN}(r)$;
            $\mathbb{R} := \texttt{RB}(def)$;
            $\mathbb{N} := \texttt{NC}(def)$;
            $A_T := \texttt{Transform}(A, C, \mathbb{R})$; $N_T := \texttt{Transform}(A, C, \mathbb{N})$;
            $r := \texttt{Normalize}([A_T, C, P, \text{and}(N, N_T)])$;
            $A' := \texttt{getI}(r)$; $C' := \texttt{getC}(r)$;
            $P' := \texttt{getP}(r)$; $N' := \texttt{getN}(r)$;
            if A 不在 A' 中出现 **then**
              $r := \texttt{Normalize}([\texttt{CRA}(A'), C', P', N'])$;
            $r := \texttt{Normalize}([\texttt{getI}(r), \texttt{CRA}(\texttt{getC}(r)), \texttt{getP}(r), \texttt{getN}(r)])$;
            $r := \texttt{Normalize}([\texttt{getI}(r), \texttt{getC}(r), \texttt{CRA}(\texttt{getP}(r)), \texttt{getN}(r)])$;
            $r := \texttt{Normalize}([\texttt{getI}(r), \texttt{getC}(r), \texttt{getP}(r), \texttt{CRA}(\texttt{getN}(r))])$;
            break ; // 应用多个概念时, 仅使用最靠前的一个进行化简
if I 形如 $[\mathcal{A}, \mathcal{C}, \mathcal{P}, \mathcal{N}]$ **then**
  $r := \texttt{Normalize}([\texttt{CRA}(\texttt{getI}(I)), \texttt{getC}(I), \texttt{getP}(I), \texttt{getN}(I)])$;
  $r := \texttt{Normalize}([\texttt{getI}(r), \texttt{CRA}(\texttt{getC}(r)), \texttt{getP}(r), \texttt{getN}(r)])$;
  $r := \texttt{Normalize}([\texttt{getI}(r), \texttt{getC}(r), \texttt{CRA}(\texttt{getP}(r)), \texttt{getN}(r)])$;
  $r := \texttt{Normalize}([\texttt{getI}(r), \texttt{getC}(r), \texttt{getP}(r), \texttt{CRA}(\texttt{getN}(r))])$;
if I 形如 $\mathbb{V} := \mathcal{K}$ **then**
  $r := \texttt{Normalize}(\mathbb{V} := \texttt{CRA}(\mathcal{K}))$;
if I 形如 $\{\beta_1; \cdots; \beta_k\}$ **then**
  $r := \{\texttt{CRA}(\beta_1); \cdots; \texttt{CRA}(\beta_k)\}$;
else
  $r := [I,$ null, null, null$]$;
返回 r;

定理 4.4.2. 分句规约算法 (算法 3) 可终止, 输出为正规型, 并且其中包含的所有实例均在 $\mathbb{S}$ 下根据任何定义都是不可被化简的.

证明: 在算法 3 中, 由于第 6、26、32、34 行输出的都是正规化结果, 由引理 4.4.1 可知, 它们一定是正规型. 若以上结果都是正规型, 则第 36 行输出为集合正规型. 第 38 行输出的结果显然为正规型. 因此, 我们需要分析并证明算法中出现的所有正规化操作以及所有递归调用都可执行.

由定义 4.3.3 可知, 算法将从第 2 行或第 33 行开始执行.

- 若算法从第 2 行开始, 如果 I 是集合的操作实例, 则第 4 行的命令将被执行并重新调用此算法执行第 36 行的命令, 然后对每个分量重复整个分析证明过程. 否则, 第 6 行中的命令将被执行. 应用内层优先的策略, 算法将为 I 包含的所有参数构建一个栈直到参数的层数为 0. 由于 I 的参数个数以及层数都有限, 因此栈的深度也有限.

  栈中元素将会按次序弹出. 若弹出元素的层数为 0, 第 38 行的命令将被执行并返回正规型. 若弹出元素的集合层数不为 0, 第 36 行的命令将被执行. 否则, 第 2 行的命令将被执行. 在第 6 行, 正规化操作将根据 $a_i\ (1 \leqq i \leqq n)$ 的规约结果的形式来进行.

  – 若它们都是集合正规型, 则正规化操作将按照规则 (11) 或者 (12) 进行;
  – 否则, 正规化操作将按照规则 (7) 进行.

  如果第 6 行的正规化结果是集合正规型, 那么算法将被重新调用并执行第 36 行的命令, 然后对每个分量执行第 29-32 行的命令. 在第 29 行, 算法将被重新调用并通过执行第 2 行或者第 36 行的命令来规约主体. 正规化操作将按照规则 (3) 进行并且新生成的约束、语境、非退化条件将分别与当前的约束、语境、非退化条件合并. 相似地, 在第 30、31、32 行, 算法将被重新调用并且正规化操作分别按照规则 (4)、(5)、(6) 进行.

  如果第 6 行的正规化结果的集合层数为 0 并且其主体 A 是一个实例, 那么在第 12 行将应用概念匹配算法生成在脚本 $\mathbb{S}$ 下目标概念与 A 相匹配的定义的集合 $defs$. 若 A 在 $\mathbb{S}$ 下按照 $defs$ 中的定义 def 可被化简, 则将分别根据从第 16 行和第 17 行得到的模板对 A 进行转化并生成由 A 决定的约束和非退化条件. 第 19 行的正规化操作将按照规则 (1)、(2) 或者 (3) 进行. 由于主体 A' 是新生成的, 需要重新调用此算法对 A' 进行规约. 第 23 行的正规化操作将按照规则 (3) 进行. 接下来, 我们分析 A' 的形式.

  - 由于 $\mathbb{R}$ 可能是集合返回体, 因此 A' 可能是集合正规型. 这样将重新调用算法并执行第 36 行的命令;
  - 否则, A' 一定是一个变量、实例或者布尔分句. 这样将重新调用算法并执行第 38 行或者第 2 行的命令.

  接着将执行第 24-26 行的命令, 过程类似于执行第 30-32 行的命令.

  如果第 6 行的正规化结果不满足第 7 行和第 9 行的条件, 那么将直接输出此正规化结果.

- 若算法从第 33 行开始, 则将重新调用此算法执行第 2 的命令并重复以上过程. 正规化操作按照规则 (8) 或者 (13) 进行.

综上所述, 算法中出现的所有正规化操作以及递归调用都可以执行. 接下来, 我们证明算法的终止性. 由于栈的深度有限, 如果所有递归调用所操作的对象都彼此不同, 那么算法将终止.

在第 6 行, 由于层数有限, a_i $(1 \leqq i \leqq n)$ 一定都不同于 I; 在第 4、8、36 行, 由于集合层数有限, α_{i_t} $(1 \leqq t \leqq m)$、β_j $(1 \leqq j \leqq k)$ 也一定都不同于 I; 在第 24-26 行和第 29-32 行, $\texttt{getC}(r)$、$\texttt{getP}(r)$、$\texttt{getN}(r)$ 中出现的实例都不同于主体中出现的实例[注1]; 在第 34 行, $\mathcal{K}$ 一定不同于 I. 因此, 算法在这些行不会操作相同的对象从而不会陷入死循环.

在第 23 行, 由于对 A 进行规约将重新调用此算法来规约 A', 如果 A' 的规约结果的主体和 A 相同, 那么算法将陷入死循环从而不会终止. 我们接下来证明这种情形不会发生.

因为 A 是 r 的主体并且 $\texttt{SL}(r) = 0$, A 在 $\mathbb{S}$ 下按照 def 是可被化简的, A 一定应用了 def 的目标概念.

- 若 $\mathbb{R}$ 是约束绑定, 则 A' 可能是一个新实例并且 $\texttt{Type}(A')$ = $\texttt{Type}(\texttt{CI}(\mathbb{R}))$. 由定义 4.4.5 可知, $\texttt{Type}(A) \prec \texttt{Type}(A')$;
- 若 A' 是复合布尔分句, 对 A' 的规约 (在第 6 行) 转化为对每个实例参数的规约. 由定义 4.4.5 可知, 这些参数的类型都是 $\texttt{Type}(A)$ 的父类型;
- 若 $\mathbb{R}$ 是集合返回体, 那么 A' 是集合正规型. 对 A' 的规约 (在第 36 行) 转化为对 A' 每个分量的规约, 然后 (在第 29 行) 转化为对这些分量主体的规约. 由定义 4.4.5 可知, 这些主体的类型都是 $\texttt{Type}(A)$ 的父类型.

[注1] 一般地, 定义中所有出现的实例都不是此定义目标概念的可解释实例.

因为 $\mathbb{S}$ 是良好的脚本, 由推论 4.4.1 可知, $\texttt{Type}(A)$ 和它的父类型都不相等. 因此, 在第 23 行对 A' 进行规约的过程中, 对 A 的规约将不会被重新调用, 从而算法在第 23 行一定终止.

综上所述, 算法不会陷入死循环而必然经过有限步终止.

由于输出结果的每个分量都是规约结果, 因此, 其中出现的所有在 $\mathbb{S}$ 下按照某个定义可被化简的实例都一定在第 18 行被化简. 否则, 它将会保留下来作为输出. 定理得证.

□

注 4.4.4. 由定理 4.4.2 可知, 分句规约算法输出便是分句 I 在脚本 $\mathbb{S}$ 下的规约结果, 因此算法解决了分句规约问题. 由于在规约过程中, 由 I 决定的所有约束关系与构造方式都被保留, 因此规约结果与 I 是同义 (或者等价) 的.

对于其它 GDL 表述 (如构型、命题、问题等), 可以根据其结构对所包含的分句依次应用分句规约算法, 进而得到其规约结果. 然而在实际应用当中, 通常期望将一个 GDL 表述同义转化为满足外部几何软件应用需求的表述. 因此需要根据正规型中各个分量的具体含义对规约结果进行转化. 分句的规约结果有如下两种形式:

$$[\mathcal{A},\mathcal{C},\mathcal{P},\mathcal{N}] \text{ 和 } \{\alpha_1;\cdots;\alpha_n\}.$$

定义 4.4.13. *整型*是指依次按照如下规则将正规型及其所包含的各个分量或参数 $\mathcal{F}$ 转化为其它表述的一系列操作, 所得结果称为正规型的*整型结果*.

(1) 若 $\mathcal{F}$ 形如 $f(\beta_1,\cdots,\beta_m)$, 其中部分但并不是所有参数为集合正规型并且 f 是某个概念的词汇, 那么需要根据每个参数的含义交互式地进行去除 { } 的操作;

(2) 若 $\mathcal{F}$ 是形如 $[\mathcal{A},\mathcal{C},\mathcal{P},\mathcal{N}]$ 的正规型, $\mathcal{A}$ 是一个分句, 则整型可表示为

$$[\mathcal{A},\mathcal{C},\mathcal{P},\mathcal{N}] \mapsto [\mathcal{P}+\text{configuration}(\mathcal{A}),\mathcal{C},\text{null},\mathcal{N}];$$

(3) 若 $\mathcal{F}$ 是形如 $\{\alpha_1;\cdots;\alpha_n\}$ 的正规型,

- 如果 $\mathcal{F}$ 是互斥正规型, 那么整型可表示为

$$\{\alpha_1;\cdots;\alpha_n\}\mapsto \text{or}(\alpha_1,\cdots,\alpha_n);$$

- 如果 $\mathcal{F}$ 是共存正规型, 那么整型可表示为

$$\{\alpha_1;\cdots;\alpha_n\}\mapsto \text{configuration}(\alpha_1,\cdots,\alpha_n).$$

定义 4.4.14. 给定一个构型、命题或者问题的规约结果的整型结果 $\mathcal{R}$, *去正规型*是指依次按照如下规则将 $\mathcal{R}$ 转化为不含有正规型的表述的一系列操作.

(1) 若 $\mathcal{R}$ 形如 configuration$(\alpha_1,\cdots,\text{or}(\beta_1,\cdots,\beta_m),\cdots,\alpha_n)$, 则去正规型操作将 $\mathcal{R}$ 拆分为 configuration$(\alpha_1,\cdots,\beta_1,\cdots,\alpha_n)$, configuration$(\alpha_1,\cdots,\beta_2,\cdots,\alpha_n)$, $\cdots$, 或 configuration$(\alpha_1,\cdots,\beta_m,\cdots,\alpha_n)$;

(2) 若 $\mathcal{R}$ 形如 configuration$(\alpha'_1,\cdots,\alpha'_n)$, 则去正规型操作可表示为

$$\text{configuration}(\alpha'_1,\cdots,\alpha'_n)\mapsto\text{configuration}(\texttt{getI}(\alpha'_1),\cdots,\texttt{getI}(\alpha'_n),\texttt{getC}(\alpha'_1),\cdots,\texttt{getC}(\alpha'_n),\texttt{getN}(\alpha'_1),\cdots,\texttt{getN}(\alpha'_n));$$

(3) 若 $\mathcal{R}$ 形如 $f(N,T,\mathcal{H},\alpha_1 <=> \alpha_2)$, 其中 f 是 Proposition 或者 Problem, 则去正规型操作将 $\mathcal{R}$ 拆分为 $f(N,T,\mathcal{H}+\alpha_1,\alpha_2)$ 和 $f(N,T,\mathcal{H}+\alpha_2,\alpha_1)$;

(4) 若 $\mathcal{R}$ 形如 $f(N,T,\mathcal{H}',[\mathcal{A},\mathcal{C},\text{null},\mathcal{N}])$, 其中 f 是 Proposition 或者 Problem, 则去正规型操作可表示为

- 若 $\mathcal{A}$ 是词汇为"is"的概念的实例, 则

$$f(N,T,\mathcal{H}',[\mathcal{A},\mathcal{C},\text{null},\mathcal{N}])\mapsto f(N,T,\mathcal{H}'+\mathcal{N},\mathcal{C});$$

- 否则, $f(N,T,\mathcal{H}',[\mathcal{A},\mathcal{C},\text{null},\mathcal{N}])\mapsto f(N,T,\mathcal{H}'+\mathcal{C}+\mathcal{N},\mathcal{A})$.

注 4.4.5. 几何构型、命题、问题在某个脚本下的规约结果经过整型和去正规型操作后仍然分别是几何构型、命题、问题, 所得到的表述称为在这个脚本下的*同义转化结果*. 其中的实例或者应用了基本概念, 或者在这个脚本下是不可转化的. 一个表述的同义转化结果与此表述是同义的.

GDL 表述的规约过程模拟人们对几何表述进行同义的概念替换过程, 通过复杂几何概念的定义将其等价地替换成简单几何概念, 从而达到易于操作处理的目的. 对于同一个 GDL 表述, 使用不同的脚本, 得到的规约结果也不相同, 因此在实际应用当中, 需要根据应用的具体需求来构建脚本, 然后再应用上述算法自动地对几何表述进行规约、整型、去正规型操作, 最后得到满足应用需求的几何表述, 再以此为基础开发与外部几何软件的接口从而实现定理自动证明与图形自动绘制的功能.

4.4.3 关系挖掘

如第 4.1 节所述, 教科书所包含的知识内容之间存在着各种不同的关系, 而这些关

系对于安排教科书的叙述结构以及组织教科书的内容都起着重要的作用, 对教科书的创建是不可或缺的, 并应该作为元知识数据存储到教科书知识库中. 这些关系的获取可以通过两种方式进行. 一种方式是人工地添加, 即在创建目标对象的同时, 通过引用 `objectID` 值来创建相应的链接. 另一种方式是根据几何知识的 GDL 表述来自动发现知识对象之间的关系. 几何表述所应用的几何概念的语义是通过其定义给定的, 因此这些概念的定义对象与这个几何表述所对应的知识对象之间构成了语境关系; 另外, 概念之间存在着继承关系, 这种关系蕴含在概念的定义之中, 即定义的目标概念与定义返回体本身或者其中心实例所应用的概念 (或本原和抽象概念) 之间构成继承关系, 因此这两个概念分别对应的定义对象之间也构成继承关系 (参见第 3.2.3 节对这两个关系类型的解释). 这两种关系类型都与定义对象有关, 可以应用概念匹配算法来自动获得相应的链接.

语境关系的自动发现

问题 4.4.4. 给定一个分句 I (包括集合分句) 和一个良好的脚本 $\mathbb{S}$, 求为 I 提供语境的定义的集合.

算法 4: 语境关系发现算法 (CRD)

输入: 分句 I, 良好的脚本 $\mathbb{S}$
输出: 为 I 提供语境的定义的集合

```
result := null;
I := getI(Normalize(I));
if I 形如 f(a_1, ··· , a_m) then
    if f 为 and 或者 or 或者 not 或者 give 或者 declare then
        for i ← 1 to m do
            将 CRD(a_i, 𝕊) 不重复地添加到 result中;
    else
        将 CMA(I, 𝕊) 不重复地添加到 result 中;
else
    if I 形如 V := 𝒦 then
        将 CRD(𝒦, 𝕊) 不重复地添加到 result 中;
    else
        if I 形如 {β_1, ··· , β_k} then
            for j ← 1 to k do
                将 CRD(β_j, 𝕊) 不重复地添加到 result 中;
返回 result;
```

注 4.4.6. 对于几何知识的 GDL 表述, 可以根据其结构遍历所包含的分句, 并应用算法 4 获得为其提供语境的定义的集合, 从而自动生成相应的知识对象之间语境关系的链接.

继承关系的自动发现

问题 4.4.5. 给定一个定义 $\mathbb{D}$ 和一个良好的脚本 $\mathbb{S}$, 求与 $\mathbb{D}$ 构成继承关系的链接的集合.

算法 5: 继承关系发现算法 (IRD)

输入: 定义 $\mathbb{D}$, $\mathbb{D}$ 的返回体 $\mathbb{R}$, 良好的脚本 $\mathbb{S}$
输出: 与 $\mathbb{D}$ 构成继承关系的链接的集合

$result :=$ null;
if $\mathbb{R}$ 形如 $\{\alpha_1, \cdots, \alpha_n\}$ **then**
  if $\mathbb{R}$ 为互斥集合返回体 **then**
    for $i \leftarrow 1$ **to** n **do**
      $I :=$ CI(α_i);
      $defs :=$ CMA(I);
      for def in $defs$ **do**
        将链接 $def \rightarrow_{\text{inherit}} \mathbb{D}$ 不重复地添加到 $result$ 中; // $\mathbb{D}$ 是分情形定义
        break;
  if $\mathbb{R}$ 为共存集合返回体 **then**
    for $j \leftarrow 1$ **to** n **do**
      将 IRD($\mathbb{D}, \alpha_j, \mathbb{S}$) 不重复地添加到 $result$ 中;
if $\mathbb{R}$ 形如 $f(\beta_1, \cdots, \beta_m)$ **then**
  if f 为 and 或者 not 或者 configuration **then**
    for $i \leftarrow 1$ **to** m **do**
      将 IRD($\mathbb{D}, \beta_i, \mathbb{S}$) 不重复地添加到 $result$ 中;
  if f 为 or **then**
    for $i \leftarrow 1$ **to** m **do**
      $defs :=$ CMA(β_i);
      for def in $defs$ **do**
        将链接 $def \rightarrow_{\text{inherit}} \mathbb{D}$ 不重复地添加到 $result$ 中; // $\mathbb{D}$ 是分情形定义
        break;
  else
    $defs :=$ CMA($\mathbb{R}$);
    for def in $defs$ **do**
      将链接 $\mathbb{D} \rightarrow_{\text{inherit}} def$ 不重复地添加到 $result$ 中;
      break;
if $\mathbb{R}$ 形如 $V :: T$ **then**
  将链接 $\mathbb{D} \rightarrow_{\text{inherit}} T$ 不重复地添加到 $result$ 中;
else
  if R 形如 [I where $\mathbb{C}$] **then**
    if I 形如 $V :: T$ **then**
      将链接 $\mathbb{D} \rightarrow_{\text{inherit}} T$ 不重复地添加到 $result$ 中;
    else
      将 IRD($\mathbb{D}, I, \mathbb{S}$) 不重复地添加到 $result$ 中;
  else
    if $\mathbb{R}$ 形如 [V] **then**
      将链接 $\mathbb{D} \rightarrow_{\text{inherit}}$ Type(V) 不重复地添加到 $result$ 中;
返回 $result$;

注 4.4.7. 应用算法 5 可以自动发现定义所蕴含的定义对象之间的继承关系. 知识对象之间的链接是知识图的一部分, 因此可以在教科书的创建过程中存储到教科书知识库中.

4.4.4　结构一致性检测

从外观的角度看, 传统的教科书按照从前到后的顺序将领域知识内容呈现给读者. 因此, 在电子几何教科书系统中, 我们将教科书看作是封装了相应内容的目标对象的线性排列. 令 $\mathfrak{T}$ 表示教科书所包含的目标对象的集合, 可以在 $\mathfrak{T}$ 上引入二元关系 $\lhd$ 来表示其中元素在教课书中的排列顺序. 对 $\mathfrak{T}$ 中的两个元素 a 和 b, $a \lhd b$ 表示在教科书中 a 被安排到或者放到 b 之前. 在这个意义下, 对 $\mathfrak{T}$ 中的任意元素 a、b 和 c, 有如下性质: (i) $\neg(a \lhd a)$; (ii) 若 $a \lhd b$ 且 $b \lhd c$, 则 $a \lhd c$; (iii) $a \lhd b$ 或者 $b \lhd a$. 因此, $\lhd$ 是集合 $\mathfrak{T}$ 上的严格全序, 教课书则可以表示为链 $(\mathfrak{T}, \lhd)$.

尽管对于数学文档的编写创建存在不同的风格, 但教科书中的领域知识需要以一种合理的适于学习的方式来组织. 换言之, 教科书内容需要循序渐进地、带有启发式地、系统地呈现给读者. 获取教科书所包含的目标对象之间的依赖关系十分重要, 因为正是这些依赖关系暗示了这些对象所期望出现的相对位置, 从而决定了教科书的叙述结构. 令 $\mathfrak{N}$ 表示目标对象的集合, 可以在 $\mathfrak{N}$ 上引入另一个二元关系 $\rightarrow$ 来表示其元素之间的依赖关系. 对于 $\mathfrak{N}$ 中的两个元素 A 和 B, $A \rightarrow B$ 表示在教科书中 A 应该或者期望被安排到 B 之前. 例如, `foot definition` $\rightarrow$ `Simson's theorem` 意味着垂足的定义需要在给出西门松定理之前被介绍. 在这个意义下, 对 $\mathfrak{N}$ 中任意的 A、B 和 C, 有如下性质: (i) $\neg(A \rightarrow A)$; (ii) 若 $A \rightarrow B$, 则 $\neg(B \rightarrow A)$; (iii) 若 $A \rightarrow B$ 且 $B \rightarrow C$, 则 $A \rightarrow C$. 因此, $\rightarrow$ 是集合 $\mathfrak{N}$ 上的严格 (非自反) 偏序, 相应的有向无环图 $(\mathfrak{N}, \rightarrow)$ 被称为*依赖图*.

定义 4.4.15. 令 $(\mathfrak{N}, \rightarrow)$ 是一个依赖图, 一个教科书 $(\mathfrak{T}, \lhd)$ 被称为在 $(\mathfrak{N}, \rightarrow)$ 下是*结构不一致的*, 如果

(1) $\mathfrak{T} \subseteq \mathfrak{N}$, 并且

(2) $\exists\, a, b \in \mathfrak{T}$, 满足 $a \rightarrow b$ 和 $b \lhd a$. 有序对 (b, a) 被称为教科书的*不一致实例*.

否则, 教科书称为在 $(\mathfrak{N}, \rightarrow)$ 下是*结构一致的*.

Kamareddine F. 等人 [72] 提出了一个标注系统以及相应的本体来辅助人们为数学文档添加注释, 被注释文本的逻辑结构图 (即我们所称的依赖图) 可以被自动获取和分析.

OMDoc 通过知识结构 (即我们所称的依赖图) 和叙述结构两种方式来标注课件的结构, 并举例说明了这两种结构之间的相互作用 (见 [75] 第 8 章). 另外, 通过构建可以刻画学习对象教学目的的本体 [122], 人们研究提出了针对不同的学习目的以及学生能力水平的结构化课程自适应生成技术和方法 [123]. 尽管如此, 我们所关注的不仅是怎样获得依赖图, 而且还有怎样利用它来判断教课书的呈现结构是否合理或者说适宜, 即教科书是否在这个依赖图下是结构一致的.

事实上, 由知识图的链接可以诱导出相应节点之间的依赖关系, 如表 4.1 所示 (其中"–"表示依赖不存在). 令 $(\mathfrak{N}, \mathscr{L})$ 表示 $\mathfrak{N}$ 上的一个知识图, 可以由 $(\mathfrak{N}, \mathscr{L})$ 诱导一个偏序集 $(\mathfrak{N}, \rightarrow)$, 其中 $\rightarrow$ 被定义为

$$\{(a, b) \mid A \rightarrow_\gamma B \in \mathscr{L} \wedge A \rightarrow_\gamma B \text{ 依据表 4.1 中的规则被转化为 } a \rightarrow b\}.$$

然而, 这样生成的偏序集 $(\mathfrak{N}, \rightarrow)$ 并不能保证是一个依赖图, 因为在某些情形下, 它的传递闭包可能包含 $\mathfrak{N}$ 中元素的自反对[注1]. 因此, 在实际应用当中, $(\mathfrak{N}, \rightarrow)$ 需要被 (交互式或者自动地) 修改直到它是一个依赖图, 这时可称为由知识图*诱导的依赖图* (见图 4.2).

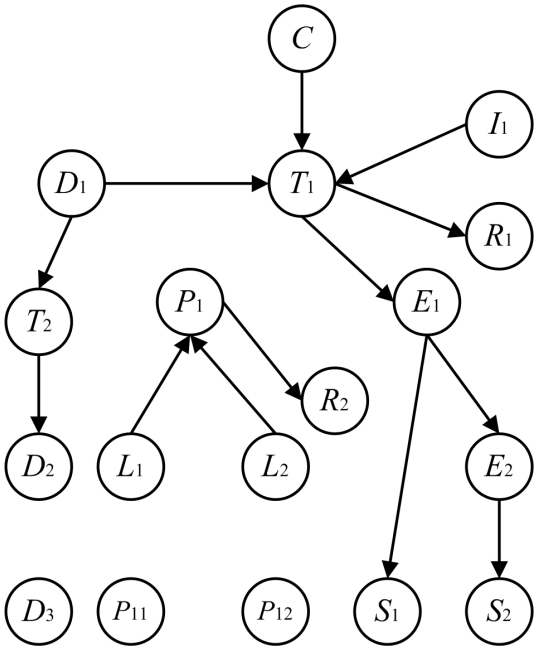

图 4.2 **由知识图 3.4 诱导的依赖图**

事实上, 教科书所包含的目标对象是教科书知识库的子集, 教科书上的知识图是教科书知识库中知识图的子图. 在构建教课书的过程中, 增加、插入、移动以及修改其中

[注1] 依赖图的判断问题本质上是有向无环图的判定问题, [72] 中有部分的讨论.

表 4.1 **链接到依赖的转换规则 (其中 A、B 是知识图中的节点)**

链接	依赖
$A \rightarrow_{\text{include}} B$	$A \rightarrow B$
$A \rightarrow_{\text{contextOf}} B$	$A \rightarrow B$
$A \rightarrow_{\text{inherit}} B$	$B \rightarrow A$
$A \rightarrow_{\text{deriveFrom}} B$	$B \rightarrow A$
$A \rightarrow_{\text{imply}} B$	$A \rightarrow B$
$A \rightarrow_{\text{hasProperty}} B$	$A \rightarrow B$
$A \rightarrow_{\text{decide}} B$	$B \rightarrow A$
$A \rightarrow_{\text{justify}} B$	–
$A \rightarrow_{\text{introduce}} B$	$A \rightarrow B$
$A \rightarrow_{\text{remarkOn}} B$	$B \rightarrow A$
$A \rightarrow_{\text{complicate}} B$	$B \rightarrow A$
$A \rightarrow_{\text{solve}} B$	$B \rightarrow A$
$A \rightarrow_{\text{applyOn}} B$	–
$A \rightarrow_{\text{exerciseOf}} B$	$B \rightarrow A$
$A \rightarrow_{\text{exampleOf}} B$	–
$A \leftrightarrow_{\text{associate}} B$	–
$A \leftrightarrow_{\text{equal}} B$	–

对象的操作都将会导致教科书内容和结构的动态改变. 在每一次这样的操作后, 系统应该自动检测在由教科书上的知识图诱导的依赖图下教科书是否是结构一致的, 并且将不一致实例显示给用户以便采取进一步的操作来构建结构一致的教科书. 算法 6 所展示的便是进行这种结构一致性检测的程序, 其中 O 是当前正在被操作的目标对象, $(\mathfrak{T}, \lhd)$ 是操作 O 完成后所构建的教科书, $(\mathfrak{T}, \mathscr{L})$ 是操作 O 完成后 $\mathfrak{T}$ 上的知识图.

算法 6 中出现的 P_d 表示在教科书中应该或期望安排到 O 前面的目标对象的集合; S_d 表示在教科书中应该或期望安排到 O 后面的目标对象的集合; P_t 表示在教科书中出现在 O 前面的目标对象的集合; S_t 表示在教科书中出现在 O 后面的目标对象的集合; P_h 表示在教科书中应该安排到 O 后面却放到了教科书前面的对象的集合; S_h 表示在教科书中应该安排到 O 前面却放到了教科书后面的对象的集合; P_r 表示在教科书中应该安

算法 6: 教课书 $(\mathfrak{T}, \lhd)$ 的结构一致性检测算法 (SCC)

输入: 教科书 $(\mathfrak{T}, \lhd)$, 知识图 $(\mathfrak{T}, \mathscr{L})$
输出: 不一致实例的集合

```
1  按照表 4.1 生成由(𝔗, ℒ) 诱导的偏序集 (𝔗, →);
2  (𝔗, →′) := (𝔗, →);
3  while (𝔗, →′) 不是一个依赖图 do
4      (𝔗, →′) := 修改 (𝔗, →′);
5  (𝔗, →̄′) := (𝔗, →′) 的传递闭包;
6  P_d := {x | (x, O) ∈ →̄′};
7  S_d := {y | (O, y) ∈ →̄′};
8  P_t := {z | (z, O) ∈ ◁};
9  S_t := {t | (O, t) ∈ ◁};
10 P_h := {(m, O) | m ∈ P_t 且 m ∈ S_d};
11 S_h := {(O, n) | n ∈ S_t 且 n ∈ P_d};
12 P_r := {m | m ∈ P_t 且 m ∈ P_d};
13 S_r := {n | n ∈ S_t 且 n ∈ S_d};
14 返回 P_h ∪ S_h;
```

排到 O 前面并且确实是放到了 O 前面的对象的集合; S_r 表示在教科书中应该安排到 O 后面并且确实是放到了 O 后面的对象的集合.

4.4.5 冗余性与完备性检测

对于一本教科书, 除了其结构上需要满足以上一致性的要求之外, 内容上还需要满足冗余性和完备性要求. 由于每一个知识对象都封装了相应的知识数据, 主题对象也封装了相关的信息, 因此教科书的内容完全由这些目标对象决定. 而每一个对象都被系统自动赋予了唯一的不重复的 `objectID` 值 (见第 3.3.1 节), 因此, 相同的对象将拥有相同的 `objectID` 值. 沿用前面使用的记号, 我们给出教科书冗余性与完备性的定义.

定义 4.4.16. 令 $\mathfrak{T}$ 表示教科书所包含的目标对象的集合, 对于 $\mathfrak{T}$ 任两个元素 a 和 b, 若 a 和 b 拥有不同的 `objectID` 值, 则称教科书是*非冗余的*. 否则称教科书是*冗余的*.

定义 4.4.17. 给定教科书 $(\mathfrak{T}, \lhd)$ 和 依赖图 $(\mathfrak{N}, \rightarrow)$, $(\mathfrak{N}, \rightarrow)$ 的传递闭包记为 $(\mathfrak{N}, \Rightarrow)$. 集合 $\{b \mid (b, a) \in \Rightarrow \wedge\ a \in \mathfrak{T}\}$ 称为教科书在 $(\mathfrak{N}, \rightarrow)$ 下的*闭包*, 记为 $\overline{\mathfrak{T}}_{\rightarrow}$. 若 $\overline{\mathfrak{T}}_{\rightarrow} \subseteq \mathfrak{T}$, 则称教科书在 $(\mathfrak{N}, \rightarrow)$ 下是*完备的*; 否则, 称教科书是*不完备的*, 并且 $\overline{\mathfrak{T}}_{\rightarrow} \setminus \mathfrak{T}$ 中的元素称为教科书在 $(\mathfrak{N}, \rightarrow)$ 下的*预备对象*.

基于定义 4.4.16, 教科书的冗余性检测可以通过判断教科书所包含的目标对象的 `objectID` 值是否重复来解决. 同样, 对于教科书完备性的检测, 可以遍历 $\mathfrak{T}$ 中的所有元素, 对每一个元素, 通过由教科书知识库的知识图所诱导的依赖图求得所有它所依赖的

目标对象的集合, 然后判断这些对象的 objectID 值是否与 $\mathfrak{T}$ 中某个元素的 objectID 值相同, 若都不相同则说明这个对象是预备对象, 这时的教科书是不完备的, 需要将所有预备对象提示给用户以便将它们添加到当前教课书中. 在这种机制下, 教科书冗余性和完备性的检测程序可以得到容易地设计与开发.

第 五 章　电子几何教科书系统的具体实施

5.1　几何教科书知识库的构建

电子几何教科书系统以几何知识库为平台来存储与管理教科书所包含的知识数据, 提供用户创建、维护、共享动态几何教科书的服务与功能. 一个规模庞大、数据丰富、结构合理的知识库对于电子几何教科书系统或者与几何有关的研究和应用而言都具有重大的实际意义. 几何知识库的设计方法与实现过程已在第 3 章进行了分析与讨论. 由于几何学包含的范围很广, 目前我们以欧几里德平面几何知识作为考察的对象, 从几何教科书知识数据的搜集与整理的角度来讨论并展示几何知识库的具体构建.

作为经典的几何教科书, 几何重观 (Geometry Revisited) [35] 包含有大量经典且复杂的几何定理、漂亮的证明以及关于圆、三角形、四边形、圆锥曲线等几何对象的性质. 因此, 我们选择这本教科书作为蓝本, 应用电子几何教科书系统对其进行数字化、整理、存储、组织, 并扩充了从其它资源所获得的几何知识数据, 由此构建了一个几何知识库, 取名为 GeoData. 知识数据的获取和整理工作是由作者和谯梁同学共同完成的, 所参考的书籍还有 Mechanical Geometry Theorem Proving [26] 和 Lessons in Geometry: I. Plane Geometry [58]. 目前 GeoData 可通过 `http://geo.cc4cm.org/geodata/` 进行访问、浏览及查询, 它包含 125 个定义、849 个定理、部分定理的证明以及背景介绍等知识数据, 并且对知识对象按照不同的分类规则进行了分类, 还构建了部分知识对象上的知识图 (请参见第 3.2.3 节). 我们应用第 4.3 节中所设计的几何知识的表述语言 (包括形式的几何描述语言 4.3.2、其它表述的形式语言 4.3.3 以及自然语言与 XML 混合使用的语言 4.3.4) 来表述各个目标对象所包含的内容, 并将这些数据存储到教科书知识库中. 由于这些目标对象所包含的数据最后将被转化为 XHTML 文档供阅读浏览和打印, 因此可以使用 XHTML 所使用的标签对自然语言表述的文本进行样式化.

由上一章对 GDL 表述同义转化的讨论可知, 几何表述所应用的导出概念的实例可以在脚本下规约为只应用基本概念或者不可转化的概念的实例, 因此, 在几何表述中的所有实例都可以被转化的情形下, 只需要使用一个固定的基本概念集便可以达到表述的目的. 基本概念集对基于几何表述的应用, 如开发几何表述与已有几何软件的接口, 开发几何定理证明器、动态几何软件、几何问题解决软件等有着重要的意义, 因为问题的表

述在经过规约之后仅应用这个基本概念集中的概念, 从而软件只需要实现这些基本概念便可以与用户交互并解决问题. 我们通过分析整理欧几里德平面几何中所涉及的概念的定义, 得到如下基本概念集[注1]. 其中, 将几何量进行严格分类的目的是为了有效地检测几何语句表述的合理性, 比如一个代数量 (无量纲) 与一个类型是长度的几何量的乘积仍然是一个类型为长度的几何量, 那么将它与一个类型为面积的几何量比较大小是不可行的, 而两个类型都是长度的几何量的乘积是类型为面积的几何量, 这种情况下比较大小才是有意义的. 这样的处理相当于对代数计算增加了几何语义的信息. 一方面对计算的可行性设置评估条件来保证几何知识库的合理性; 另一方面, 在实际计算时, 可以把几何量作为代数量进行各种运算操作并不会影响计算的效率, 因此有助于几何知识库与符号计算工具的兼容.

表 5.1 **表示几何对象与几何量的基本概念集**

基本概念	含义
point()	任意一个点
line(A::Point, B::Point)	过 A、B 的直线
segment(A::Point, B::Point)	以 A、B 为端点的线段
halfline(A::Point, B::Point)	以 A 为原点过 B 的射线
triangle(A::Point, B::Point, C::Point)	以 A、B、C 为顶点的三角形
quadrilateral(A::Point, B::Point, C::Point, D::Point)	以 A、B、C、D 为顶点的四边形
circle(A::Point, r::Length)	以 A 为圆心, r 为半径的圆
conic(A::Point, B::Point, C::Point, D::Point, E::Point)	过 A、B、C、D、E 的二次曲线
arc(A::Point, B::Point, C::Point)	过 A、B、C 的圆弧
angle(A::Point, B::Point, C::Point)	直线 AB 顺时针旋转到首次与直线 BC 重合所转过的角度
distance(A::Point, B::Point)	点 A 与 B 之间的距离
size(θ::Angle)	角 θ 的大小

[注1] 随着几何知识所涉及范围的扩大, 基本概念集也会相应地扩充.

表 5.2 **表示量函数的基本概念集**

基本概念	含义
ratio(a::Length, b::Length)	长度 a 与 b 的比值
ratio(a::Area, b::Area)	面积 a 与 b 的比值
ratio(a::Degree, b::Degree)	角度 a 与 b 的比值
ratio(a::AlgebraicQuantity, b::AlgebraicQuantity)	代数量 a 与 b 的比值
divide(a::GeometricQuantity, b::AlgebraicQuantity)	几何量 a 的 b 等分
divide(a::Area, b::Length)	面积与长度相除
times(a::Area, b::Area)	面积与面积的乘积
times(a::Length, b::Length)	长度与长度的乘积
times(a::length, b::Area)	长度与面积的乘积
times(a::Area, b::Length)	面积与长度的乘积
times(a::AlgebraicQuantity, b::AlgebraicQuantity)	代数量 a 与 b 的乘积
times(a::AlgebraicQuantity, b::GeometricQuantity)	代数量与几何量的乘积
times(a::GeometricQuantity, b::AlgebraicQuantity)	几何量与代数量的乘积
minus(a::Length, b::Length)	长度 a 与 b 之差
minus(a::Area, b::Area)	面积 a 与 b 之差
minus(a::Degree, b::Degree)	角度 a 与 b 之差
minus(a::AlgebraicQuantity, b::AlgebraicQuantity)	代数量 a 与 b 之差
plus(a::Length, b::Length)	长度 a 与 b 之和
plus(a::Area, b::Area)	面积 a 与 b 之和
plus(a::Degree, b::Degree)	角度 a 与 b 之和
plus(a::AlgebraicQuantity, b::AlgebraicQuantity)	代数量 a 与 b 之和
sin(θ::Angle)	角 θ 的正弦
cos(θ::Angle)	角 θ 的余弦
tg(θ::Angle)	角 θ 的正切
ctg(θ::Angle)	角 θ 的余切

表 5.3 **表示几何关系的基本概念集**

基本概念	含义
is(A::Point, B::Point)	点 A 与 B 是同一点
incident(A::Point, l::Line)	点 A 在直线 l 上
isin(A::Point, t::Triangle)	点 A 在 t 内部
perpendicular(l::Line, m::Line)	直线 l 与 m 垂直
parallel(l::Line, m::Line)	直线 l 与 m 平行
sameside(A::Point, B::Point, C::Point)	点 A、B 在点 C 的同侧
differentside(A::Point, B::Point, C::Point)	点 A、B 在点 C 的两侧
sameside(A::Point, B::Point, l::Line)	点 A、B 在直线 l 的同侧
sameside(A::Point, B::Point, l::Line)	点 A、B 在直线 l 的两侧

表 5.4 **表示量关系的基本概念集**

基本概念	含义
lt(a::Length, b::Length)	长度 a 小于长度 b
lt(a::Area, b::Area)	面积 a 小于面积 b
lt(a::Degree, b::Degree)	角度 a 小于角度 b
lt(a::AlgebraicQuantity, b::AlgebraicQuantity)	代数量 a 小于代数量 b
gt(a::Length, b::Length)	长度 a 大于长度 b
gt(a::Area, b::Area)	面积 a 大于面积 b
gt(a::Degree, b::Degree)	角度 a 大于角度 b
gt(a::AlgebraicQuantity, b::AlgebraicQuantity)	代数量 a 大于代数量 b
equal(a::Length, b::Length)	长度 a 与 b 相等
equal(a::Area, b::Area)	面积 a 与 b 相等
equal(a::Degree, b::Degree)	角度 a 与 b 相等
equal(a::AlgebraicQuantity, b::AlgebraicQuantity)	代数量 a 与 b 相等

表 5.5 几何作图的基本概念集

基本概念	含义
perpendicularline(A::Point, l::Line)	过点 A 与 l 垂直的直线
parallelline(A::Point, l::Line)	过点 A 与 l 平行的直线
pointon(o::Circle)	圆 o 上任一点
pointon(l::Line)	直线 l 上任一点
midpoint(s::Segment)	线段 s 的中点
circle(A::Point, B::Point)	以 A 为圆心过 B 的圆
circle(A::Point, B::Point, C:Point)	过 A、B、C 的圆
intersection(l::Line, m::Line)	直线 l 与 m 的交点
intersection(l::Line, o::Circle)	直线 l 与圆 o 的交点
intersection(h::Halfline, o::Circle)	射线 h 与圆 o 的交点
intersection(m::Circle, n::Circle)	两圆 m 与 n 的交点
tangentpoint(l::Line, o::Circle)	直线 l 与圆 o 的切点
tangentpoint(m::Circle, n::Circle)	两圆 m 与 n 的切点
2tangentline(A::Point, o::Circle)	过圆外一点 A 与圆 o 的切线
center(o::Circle)	圆 o 的圆心
bisectorline(A::Point, B::Point, C::Point)	直线 AB 与直线 BC 所成角的平分线
rotate(A::Point, B::Point, d::Degree)	将 A 绕 B 逆时针旋转角度 d 所得点
symmetry(A::Point, B::Point)	点 A 关于 B 的对称点

5.2 电子几何教科书系统的人机界面

清晰明确的叙述结构可以增强教科书的可读性并帮助读者不需要阅读细节便可以获知相关理论的内容. 通常, 教科书包括多个章, 而每章又包含多个节, 每节又通过多个段来呈现知识内容. 教科书的这种“章-节-段”的层次结构本质上是对相关几何知识的具体分类. 常用的教科书编写系统都是基于文本的线性编辑模式, 通过对文本进行标注来实现对结构的控制, 教科书则以文档的形式存储. 然而, 要对教科书的内容和结构进行修改则需要不断地对文本进行重新定位, 并且对教科书文档中某一部分的维护、共享与处理也不是很直观和容易的. 为了便于对教科书的结构进行控制, 我们基于第 3 章中对几

何知识数据封装和分类的思想, 将教科书的界面设计成由一系列节点构成的树形结构, 称为*目录树*, 其中每个分支节点都表示主题对象, 并显示相应的 `role` 和 `name` 值, 每个叶子节点都表示知识对象, 并显示相应的知识类名与 `name` 值, 根节点则表示教科书这个主题对象, 显示教科书的 `name` 值. 在整个结构中, 父节点与子节点之间构成包含关系, 每个节点都可通过其所存储的 `objectID` 值链接到知识库中相应的主题对象或者知识对象. 图 5.1 展示了使用电子几何教科书系统所构建的 Geometry Revisited 的目录树. 对这样的教科书结构上的操作即是对目录树的操作, 而对内容上的操作则通过节点转化为对教科书知识库中知识对象或主题对象的操作.

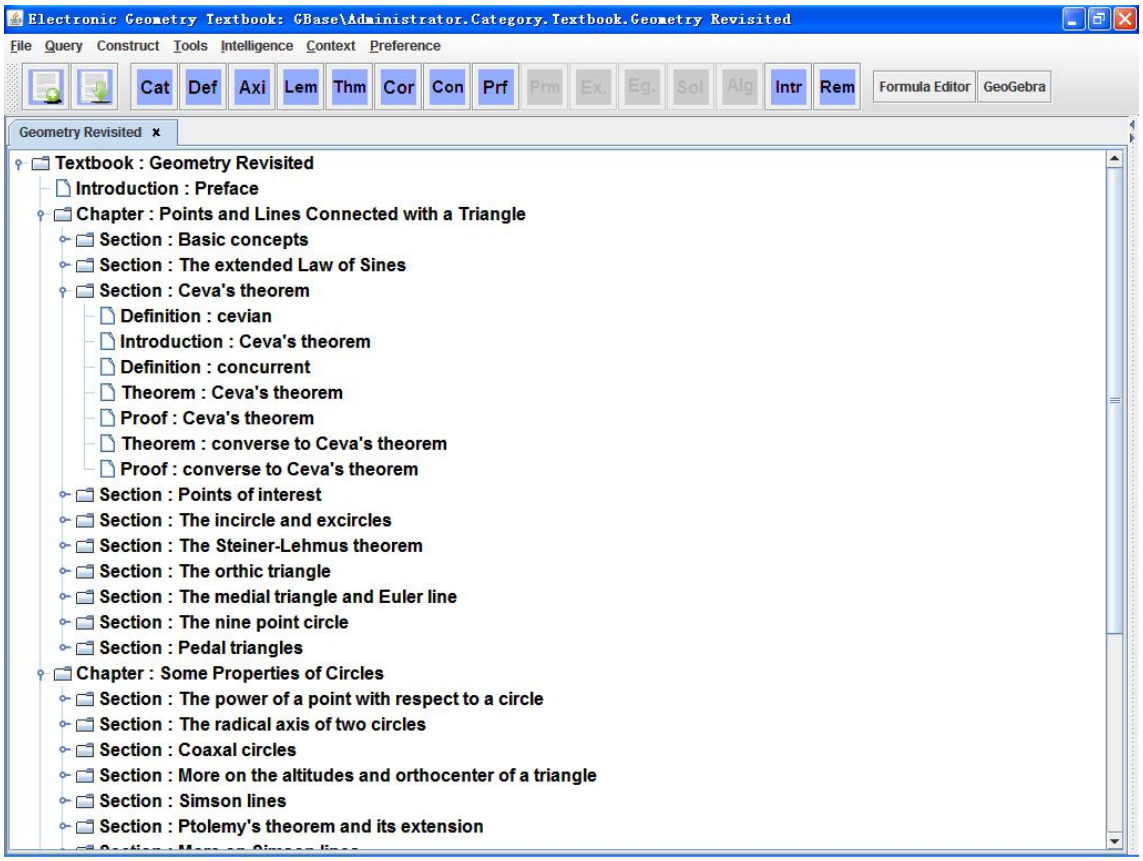

图 5.1 **电子几何教科书系统的外观**

由于教科书所包含的内容数据通过已构建的教科书知识库来存储和管理, 因此电子几何教科书系统通过调用几何知识库对目标对象的数据管理接口和对话框, 提供用户创建以及维护教科书的操作, 包括向目录树中添加、插入新建的或者检索知识库得到的节点, 修改节点链接的目标对象所封装的多版本知识数据或者相关信息, 删除节点, 拖动节点从一个位置到另一个位置来改变目录树的结构, 以及将构建完成的目录树作为类别

(即仅涉及包含关系的知识图) 存储到教科书知识库中, 以实现教科书的共享与重用.

电子几何教科书系统除了提供这些基本的功能之外, 还具有智能辅助的特性. 在构建教科书的操作过程中, 为了保证教科书的合理性, 系统需要实时地检测其结构一致性、完备性以及冗余性. 因此, 每一种操作都需要遵循一定的流程来实现实时检测的功能 (见图 5.2、5.3、5.4).

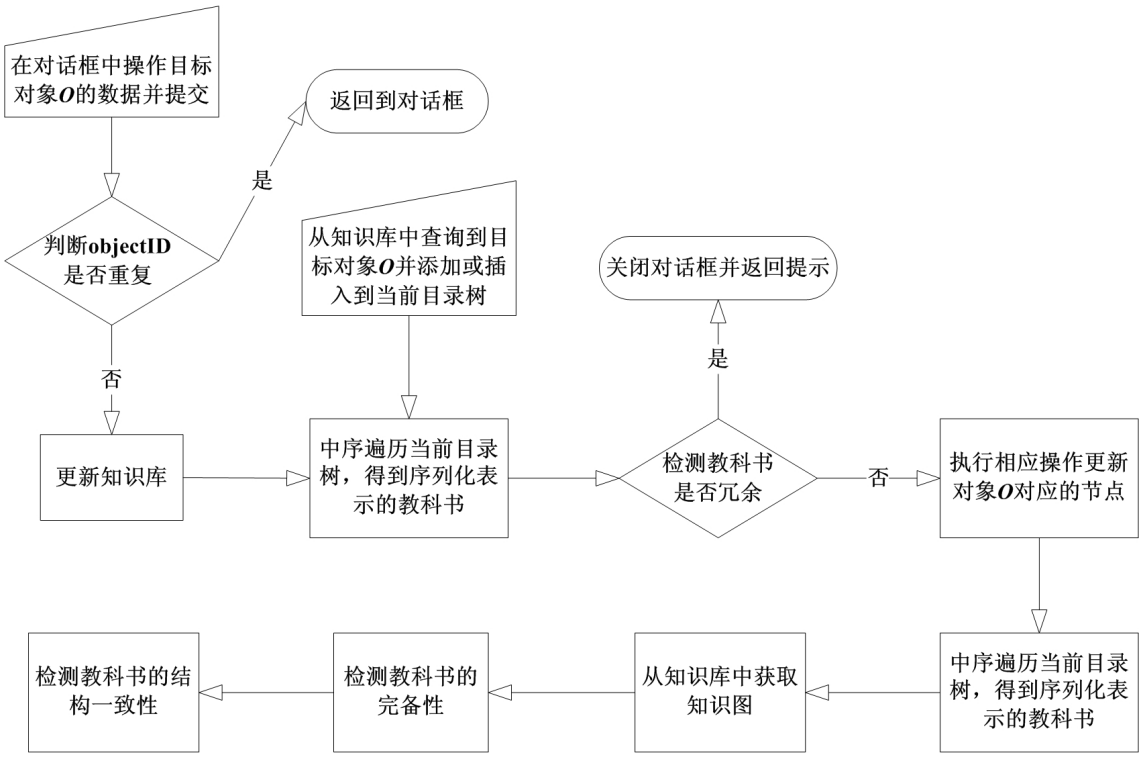

图 5.2 **添加、插入、修改目录树节点 (链接的目标对象为 *O*) 操作的流程图**

图 5.5 展示了在创建教科书 Geometry Revisited 的过程中, 当将新创建的西门松定理 (`Simson's theorem`) 插入到垂足的定义 (`foot definition`) 之前 (垂足是西门松定理表述所应用的概念), 系统将首先检测教科书的完备性, 提示用户 collinear、incident、point 这三个几何概念的定义并没有包含在当前教科书中, 同时改变垂足定义的节点颜色以提示用户应该将西门松定理节点调整到它的后面.

5.3 电子几何教科书的呈现

一方面, 教科书通过目录树与用户进行实时地交互, 从而改进其内容和结构. 另一方面, 教科书的内容可以通过传统的样式呈现, 供人阅读浏览或者打印. 根据目录树, 系统依次获取其节点所存储的 `objectID` 值, 按照用户的需求选取每个目标对象所封装的具

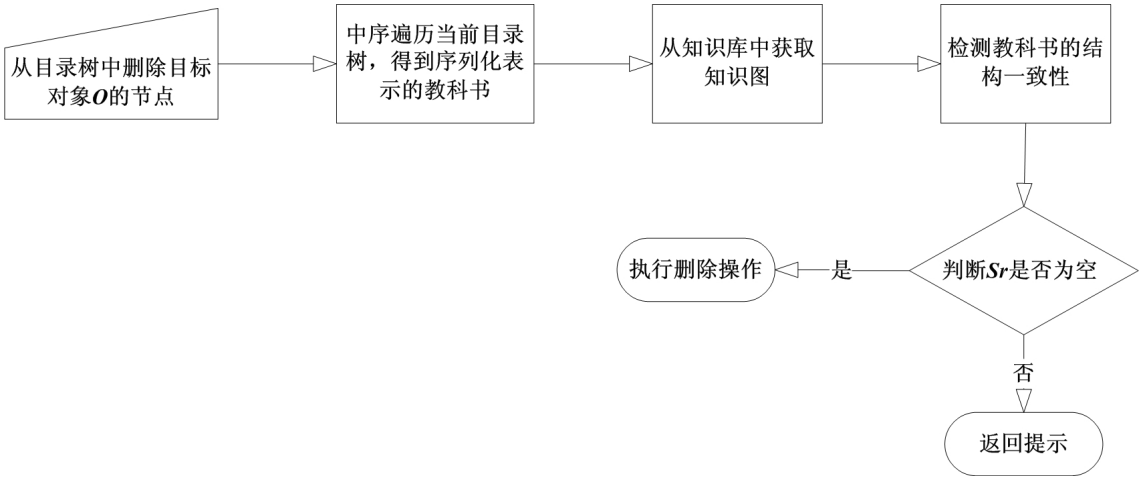

图 5.3 **删除目录树节点 (链接的目标对象为 *O*) 操作的流程图**

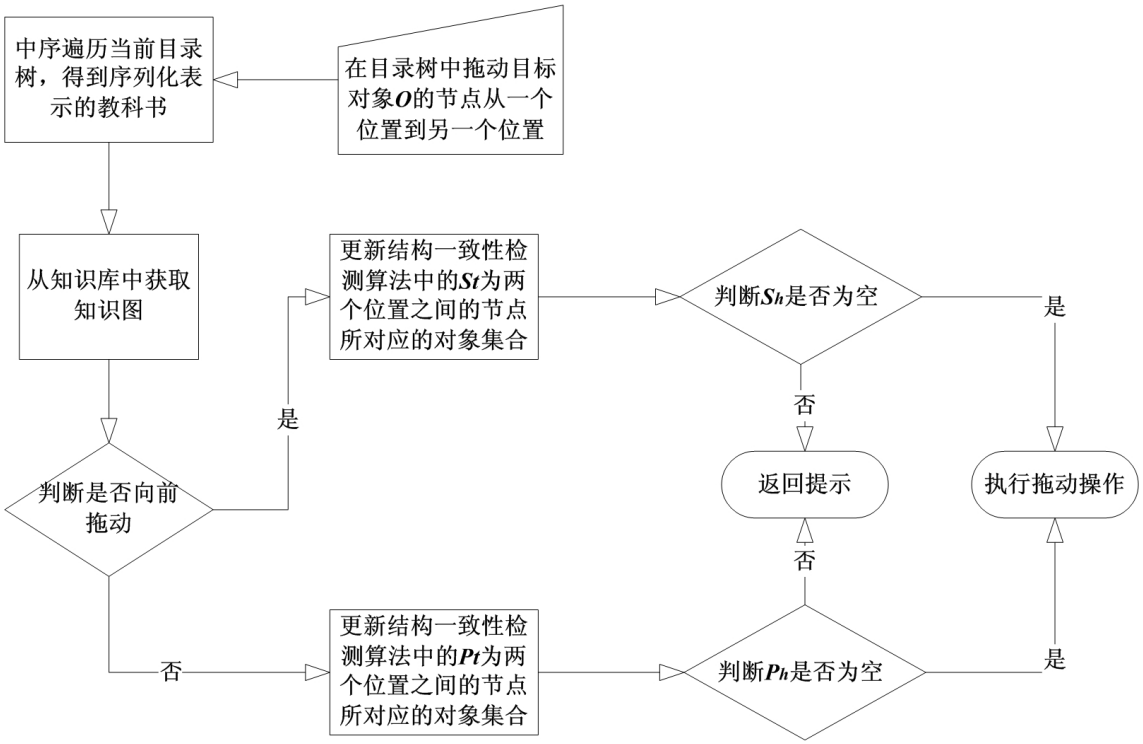

图 5.4 **拖动目录树节点 (链接的目标对象为 *O*) 操作的流程图**

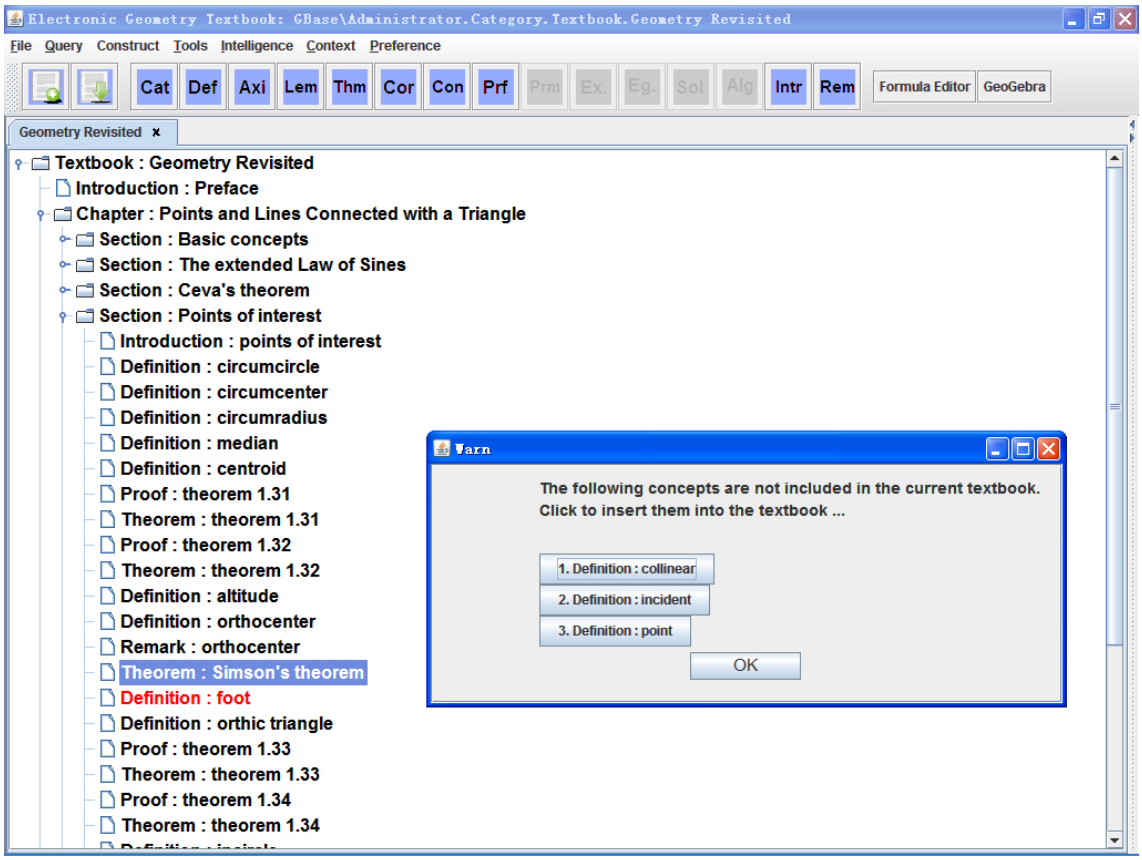

图 5.5 **对话框提示用户当前教科书不完备, 垂足定义的节点颜色改变以示教科书结构不一致**

有合适版本的数据, 调用几何知识库的浏览功能, 生成当前教科书对应的 XML 文档. 对于所生成的文档内部所有使用 GDL 表述的概念实例, 系统应用概念匹配算法 (算法 2) 获取其所应用的概念的定义, 并自动为其附上引用链接, 使得用户可以通过点击操作链接到封装此定义的知识对象在文档中的位置, 从而实现概念的实例与概念的定义之间的交叉引用. 为了将 XML 文档转化为 XHTML 文档, 我们针对其所使用的 XML 标签开发了相应的 XSLT 样式表, 并利用已有的 OpenMath 转化为 MathML 的样式表 [119] 以及 LaTeX 转化为 MathML 的样式表 [81] 实现了对数学对象的样式转化. 一篇文档中会出现多个类型相同的目标对象, 如多个节、多个定理对象等, 样式表将根据其层次结构自动生成序号, 如“Section 1.1”、“Theorem 1.2”等.

系统可以生成同一教科书的不同版本文档, 并提供了语言的选择功能方便用户实时地转换系统界面以及教科书的显示语言, 使得同时有效地创建、维护以及呈现不同语言版本的教科书成为可能 (如图 5.6 和 5.7).

教科书上的知识图指明了教科书所包含目标对象之间的相互关系, 因此, 将知识图

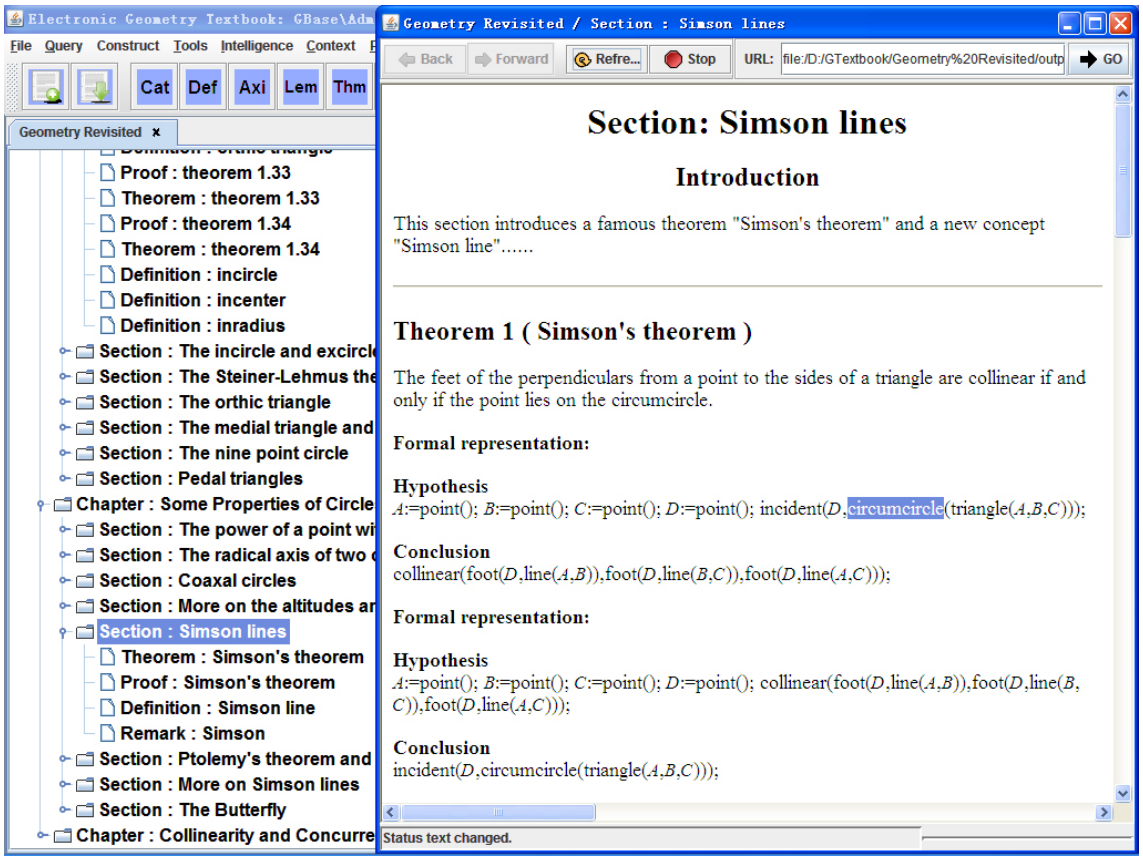

图 5.6 **西门松线一节的英文呈现**

可视化对于查看浏览教科书内容的组织结构有着重要的意义. 系统将教科书上的知识图映射到信息可视化工具包 Prefuse [108] 内部的数据结构上, 通过调用其丰富的交互式可视化功能, 动态地呈现教科书上的知识图 (见图 5.8).

5.4　几何知识的自动处理

最后我们讨论几何描述语言以及相关自动化功能的实现.

5.4.1　几何描述语言的计算机表示

对于几何描述语言, 最简单的表示方法是按照其语法规则使用文本字符串. 在构建几何知识库时, 用户可以方便地使用此种方式输入知识对象的 GDL 表述. 然而当对这些表述执行各种操作时, 如求类型、替换、化简、正规化、规约等, 需要进行大量形式上的判断, 因此 GDL 表述的结构需要明确地表示出来以方便程序的识别与处理, 这种线性的文本字符串格式不适合操作算法的实现. 我们选择 XML 作为几何描述语言的计算机

图 5.7 **西门松线一节的中文呈现**

表示语言, 基于如下考虑:

1. XML 作为一种元语言可以方便地加以定义和扩展;
2. XML 作为一种标注语言可以明确直接地将几何描述语言中的各种形式表示为树状结构, 并且通过标注获得各个部分的语义信息;
3. XML 处理器具有在多种平台上丰富的应用程序接口, 为实现几何描述语言所涉及的各种操作与算法提供方便, 并且使得这些功能可以容易地与其它程序整合;
4. XML 文档可以容易地通过样式表转换为其它格式, 使得所表述的对象可以根据需求呈现出不同的风格和样式.

我们为几何描述语言设计了一套 XML 表示的标签 (请参见附录 **D.**) 并使用 Java 语言实现了一个将几何描述语言由文本字符串格式转化为 XML 表示的解析器, 同时开发了相应的样式表将 XML 表示的 GDL 表述转化为 XHTML 文档, 以实现其在教科书中的呈现. 例如, 两条直线 l 和 m 的交点定义使用字符串表示为

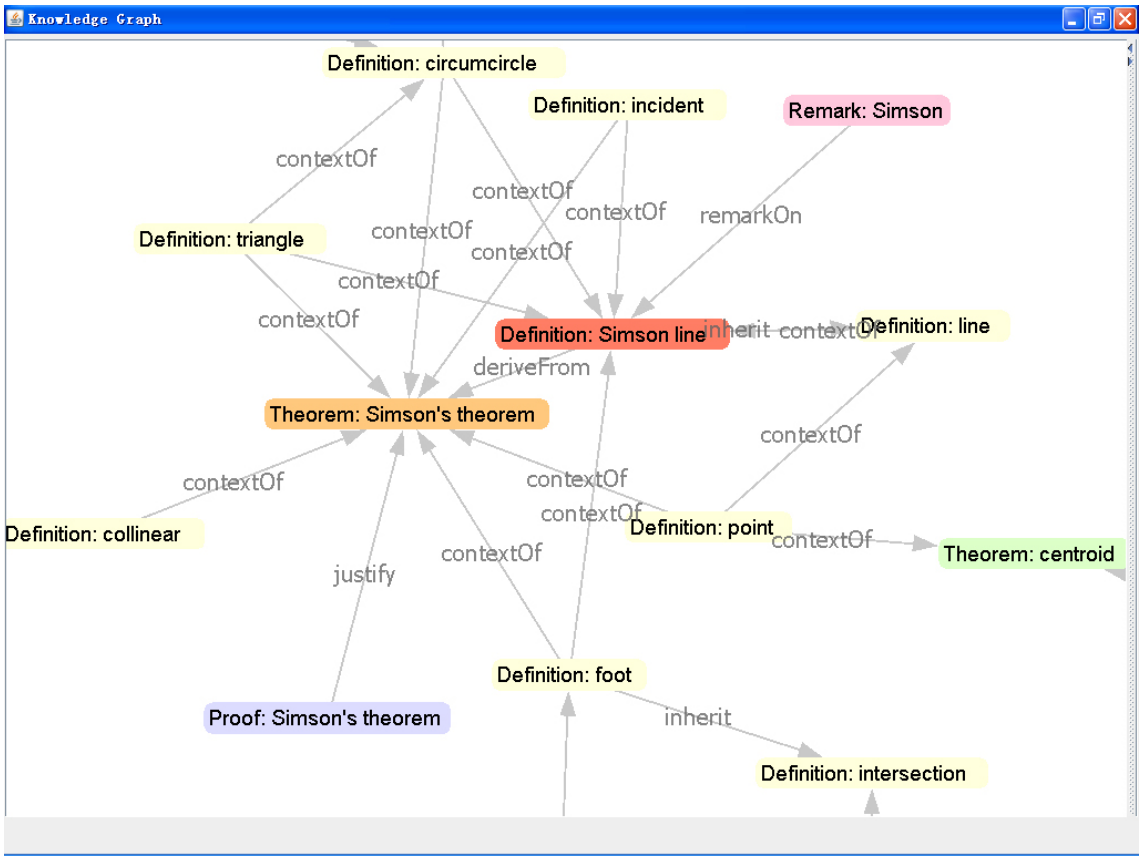

图 5.8 知识图的可视化界面

Definition(intersection(l::line, m::line), [A::point where and(incident(A, l), incident(A, m))], intersect(l, m)), 将其解析为 XML 表示如下:

```
<formalDefinition type="intersection(Line,Line)" author="Administrator" version="1">
    <conceptObject type="intersection(Line,Line)">
        <vocabulary>intersection</vocabulary>
        <argument>
            <pointer type="Line">l</pointer>
            <pointer type="Line">m</pointer>
        </argument>
    </conceptObject>
    <return count="1">
        <binding>
            <pointer type="Point" reference="new::Point">A</pointer>
            <constraint>
                <and cd="logic1" type="Boolean">
                    <conceptObject type="incident(Point,Line)">
                        <vocabulary>incident</vocabulary>
                        <argument>
                            <pointer reference="new/#1" type="Point">A</pointer>
```

```
                        <pointer reference="argument/#1" type="Line">l</pointer>
                    </argument>
                </conceptObject>
                <conceptObject type="incident(Point,Line)">
                    <vocabulary>incident</vocabulary>
                    <argument>
                        <pointer reference="new/#1" type="Point">A</pointer>
                        <pointer reference="argument/#2" type="Line">m</pointer>
                    </argument>
                </conceptObject>
            </and>
        </constraint>
    </binding>
</return>
<nondegeneracyCondition>
    <conceptObject type="intersect(Line,Line)">
        <vocabulary>intersect</vocabulary>
        <argument>
            <pointer reference="argument/#1" type="Line">l</pointer>
            <pointer reference="argument/#2" type="Line">m</pointer>
        </argument>
    </conceptObject>
</nondegeneracyCondition>
</formalDefinition>
```

西门松定理(一个分支)使用字符串表示为 Theorem(Simson's theorem, Theorem, assume (A := point(), B := point(), C := point(), D := point(), incident(D, circumcircle(triangle(A, B, C)))), show(collinear(foot(D, line(A, B)), foot(D, line(A, C)), foot(D, line(B, C))))), 将其解析为 XML 表示如下:

```
<formalRepresentation name="Simson's_theorem" type="Theorem" author="Administrator">
  <assumption>
    <reference>
      <pointer type="point()">A</pointer>
      <conceptObject type="point()" reference="Administrator_Definition_point_english.xml">
        <vocabulary>point</vocabulary>
        <argument>
        </argument>
      </conceptObject>
    </reference>
    <reference>
      <pointer type="point()">B</pointer>
      <conceptObject type="point()" reference="Administrator_Definition_point_english.xml">
        <vocabulary>point</vocabulary>
```

```
      <argument>
      </argument>
    </conceptObject>
  </reference>
  <reference>
    <pointer type="point()">C</pointer>
    <conceptObject type="point()" reference="Administrator_Definition_point_english.xml">
      <vocabulary>point</vocabulary>
      <argument>
      </argument>
    </conceptObject>
  </reference>
  <reference>
    <pointer type="point()">D</pointer>
    <conceptObject type="point()" reference="Administrator_Definition_point_english.xml">
      <vocabulary>point</vocabulary>
      <argument>
      </argument>
    </conceptObject>
  </reference>
  <conceptObject type="incident(point(),circumcircle(triangle(point(),point(),point())))"
                 reference="Administrator_Definition_incident_english.xml">
    <vocabulary>incident</vocabulary>
    <argument>
      <pointer type="point()">D</pointer>
      <conceptObject type="circumcircle(triangle(point(),point(),point()))"
                reference="Administrator_Definition_circumcircle_english.xml">
        <vocabulary>circumcircle</vocabulary>
        <argument>
          <conceptObject type="triangle(point(),point(),point())"
                  reference="Administrator_Definition_triangle_english.xml">
            <vocabulary>triangle</vocabulary>
            <argument>
              <pointer type="point()">A</pointer>
              <pointer type="point()">B</pointer>
              <pointer type="point()">C</pointer>
            </argument>
          </conceptObject>
        </argument>
      </conceptObject>
    </argument>
  </conceptObject>
</assumption>
<objective>
```

```
    <conceptObject type="collinear(foot(point(),line(point(),point())),foot(point(),
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣line(point(),point())),foot(point(),line(point(),point())))"
                    reference="Administrator_Definition_collinear_english.xml">
      <vocabulary>collinear</vocabulary>
      <argument>
        <conceptObject type="foot(point(),line(point(),point()))"
                       reference="Administrator_Definition_foot_english.xml">
          <vocabulary>foot</vocabulary>
          <argument>
            <pointer type="point()">D</pointer>
            <conceptObject type="line(point(),point())"
                       reference="Administrator_Definition_line_english.xml">
              <vocabulary>line</vocabulary>
              <argument>
                <pointer type="point()">A</pointer>
                <pointer type="point()">B</pointer>
              </argument>
            </conceptObject>
          </argument>
        </conceptObject>
        <conceptObject type="foot(point(),line(point(),point()))"
                       reference="Administrator_Definition_foot_english.xml">
          <vocabulary>foot</vocabulary>
          <argument>
            <pointer type="point()">D</pointer>
            <conceptObject type="line(point(),point())"
                    reference="Administrator_Definition_line_english.xml">
              <vocabulary>line</vocabulary>
              <argument>
                <pointer type="point()">B</pointer>
                <pointer type="point()">C</pointer>
              </argument>
            </conceptObject>
          </argument>
        </conceptObject>
        <conceptObject type="foot(point(),line(point(),point()))"
                    reference="Administrator_Definition_foot_english.xml">
          <vocabulary>foot</vocabulary>
          <argument>
            <pointer type="point()">D</pointer>
            <conceptObject type="line(point(),point())"
                    reference="Administrator_Definition_line_english.xml">
              <vocabulary>line</vocabulary>
              <argument>
```

```
            <pointer type="point()">A</pointer>
            <pointer type="point()">C</pointer>
          </argument>
        </conceptObject>
      </argument>
    </conceptObject>
  </argument>
 </conceptObject>
</objective>
</formalRepresentation>
```

5.4.2　关系的自动发现

在构建几何知识库的过程中, 目标对象之间的链接除了可以在对话框中通过人为标注的方式创建之外, 系统实现了自动发现知识对象之间的语境关系以及定义对象之间的继承关系的算法. 具体的过程是首先获取当前知识对象所封装的 GDL 表述, 然后根据其结构应用关系自动发现算法 (请参见算法 4、5) 求得与其构成相应关系的知识对象的集合或链接的集合, 最后将发现的链接输出到对话框以提示用户选择, 所选链接将存储到教科书知识库中. 例如, 在构建西门松线的定义对象时, 系统自动发现与其构成语境关系和继承关系的定义对象 (见图 5.9).

5.4.3　自动转化

为了确保教科书中几何定理等命题和证明问题的正确性, 系统提供了其 GDL 表述与外部几何定理证明器和动态几何软件的接口: 一方面可以自动构造并绘制它们对应的动态几何图形, 使得用户可以直观地判断图形所蕴含的几何性质; 另一方面通过自动证明的方式来辅助精确地判断其正确性. 由于教科书中几何命题和证明问题的表述使用了大量教科书中所定义的几何概念, 而这些概念通常是外部几何软件没有实现而不可识别和操作的. 因此, 我们将接口的实现分两步来进行. 第一步, 判断教科书的脚本[注 1] 是否是良好的. 如果不是, 则需要人为地修改脚本以使得它是良好的; 然后按照第 4.4.2 节中的方法将几何命题或者证明问题的 GDL 表述在所得脚本下进行规约, 如果规约结果中含有不可转化的概念的实例, 则在脚本中添加新的定义以化简此实例, 并重新进行这一步, 直到规约结果中的实例都是基本概念的实例; 再对规约结果进行整型和去正规型操作得

[注 1] 教科书知识库中所包含的定义对象都封装有形式化的 GDL 表述, 因此由这些表述可以生成一个脚本, 我们称其为*教科书的脚本*.

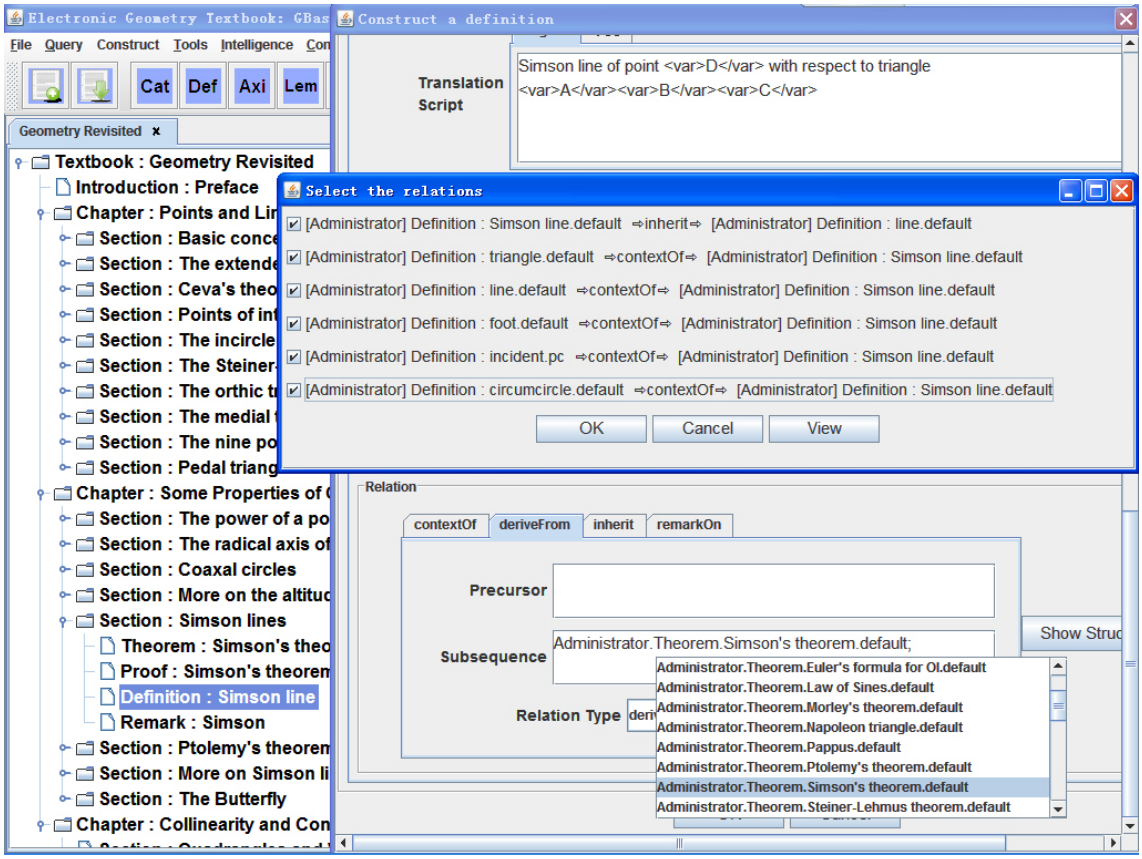

图 5.9 **自动发现语境关系和继承关系以及人工标注目标对象之间的关系**

到几何命题或证明问题的同义转化结果. 第二步, 由于基本概念集中的概念是固定的, 因此将上一步得到的表述映射为外部几何软件的内部语言表述, 进而在外部几何软件包内进行自动证明与自动作图, 最后将结果整理成文档返回到电子几何教科书系统. 具体的思想可见图 5.10.

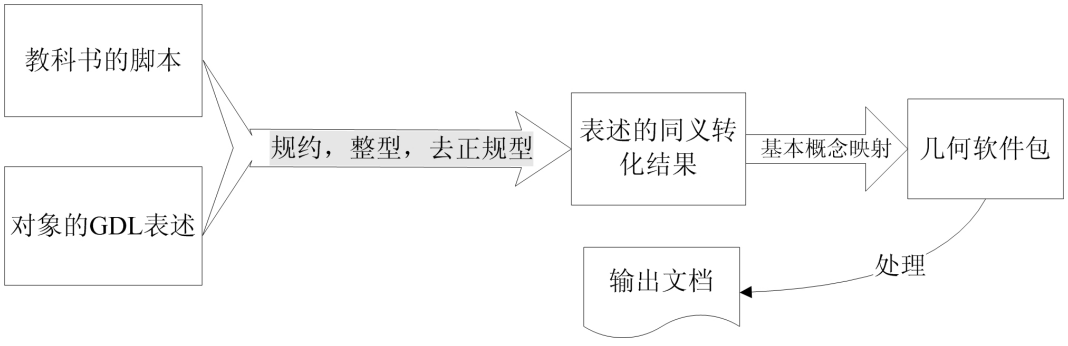

图 5.10 **电子几何教科书系统与外部几何软件包的交互流程**

系统通过检索合并教科书知识库中所有定义对象封装的 GDL 表述来自动生成教科

书的脚本. 基于 XML 表示, 我们使用 Java 的 DOM 接口实现了 GDL 表述同义转化操作的相关算法 (请参见第 4.4.2 节). 在脚本是良好的条件下, 可以将几何命题或证明问题的 GDL 表述自动规约为只应用相应的基本概念集中的概念的表述. 由此, 同义转化结果可以通过简单的转换程序映射到外部几何软件内部表示.

5.4.4　自动证明

我们选择应用坐标代数法自动证明几何定理的软件包 GEOTHER [127] 作为电子几何教科书系统的外部证明器. 由于此证明器可以识别并处理的语句为声明语句或者布尔语句, 因此我们需要将概念的定义表述为约束型定义, 相应的脚本可参见附录 **E.** 同义转化结果应用了表 5.1、5.2、5.3、5.4 所包含的概念. 我们应用 Java 语言开发了转换程序将应用基本概念的 GDL 表述转换为 GEOTHER 的内部表示, 并应用 Java OpenMaple 实现了调用 GEOTHER 的接口, 具体过程如下. 首先将应用基本概念的分句映射为 GEOTHER 中的谓词表达式 (见表 5.6), 然后再将整个表述结构按如下规则进行转化 Proposition$(N, T, \mathcal{H}, \mathcal{C})$[注1] $\mapsto N :=$ Theorem$([\alpha_1, \cdots, \alpha_n], [\beta_1, \cdots, \beta_m])$, 其中 α_i $(1 \leqq i \leqq n)$ 和 β_j $(1 \leqq j \leqq m)$ 分别为 $\mathcal{H}$ 和 $\mathcal{C}$ 中的分句经过映射后得到的谓词表达式.

表 5.6　**将 GDL 分句映射为 GEOTHER 中的谓词表达式**

分句	谓词表达式
A:=point()	arbitrary(A)
declare(A::Point)	arbitrary(A)
is(A, B)	equal(A, B)
equal(distance(A, B), distance(C, D))	equidistance(A, B, C, D)
equal(length(side(A, B)), length(side(C, D)))	equidistance(A, B, C, D)
equal(size(angle(A, B, C)), size(angle(D, E, F)))	equiangle(A, B, C, D, E, F)
incident$(A$, line$(B, C))$	collinear(A, B, C)
incident$(A$, segment$(B, C))$	collinear(A, B, C)
incident$(A$, circle$(B$, distance$(C, D)))$	equidistance(A, B, C, D)
perpendicular(line(A, B), line(C, D))	perpendicular(A, B, C, D)
parallel(line(A, B), line(C, D))	parallel(A, B, C, D)

[注1] 此处也可以是 Problem$(N$, Prove$, \mathcal{H}, \mathcal{C})$.

表 5.6 将 GDL 分句映射为 GEOTHER 中的谓词表达式 (续)

分句	谓词表达式
equal(times(distance$(A,B),m),n)$	sdistance$(A,B)*m^2-n^2$
area(triangle(A,B,C))	area(A,B,C)
tg(angle(A,B,C))	tg(A,B,C)

目前, 我们使用 GEOTHER 对 GeoData 所存储的一部分定理进行了自动证明, 其余的测试工作仍在进行当中. 图 5.11 展示了应用 GEOTHER 自动证明教科书中的西门松定理, 并将证明结果通过文档输出到电子几何教科书系统的界面中.

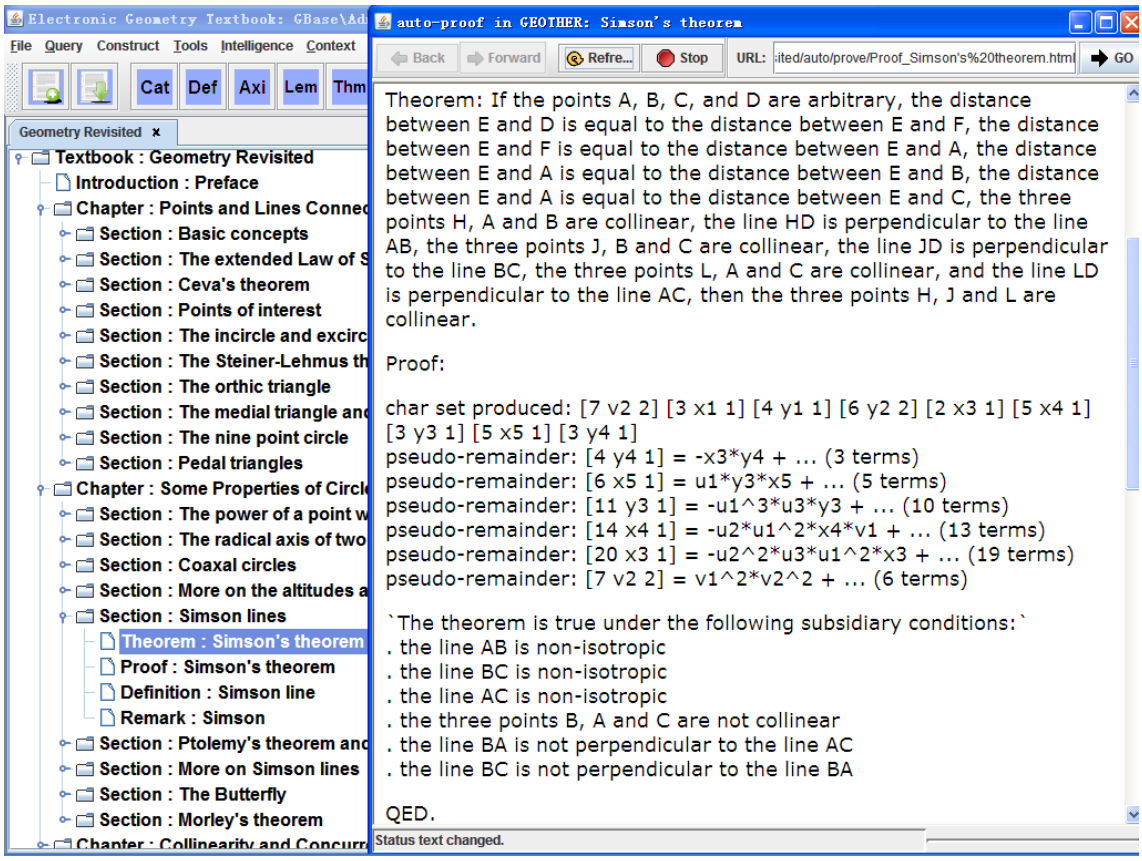

图 5.11 调用 GEOTHER 自动证明西门松定理

5.4.5 自动作图

我们选择动态几何软件 GeoGebra [46] 作为电子几何教科书系统自动绘制动态几何图形的外部工具. 由于此软件可以识别并处理的语句为声明语句或者引用语句, 因

此我们需要将概念的定义表述为构造型定义, 相应的脚本可参见附录 **F.** 同义转化结果应用了表 5.1、5.2、5.5、5.4 所包含的概念. 我们应用 Java 语言开发了转换程序将应用基本概念的 GDL 表述转换为 GeoGebra 的内部表示, 并自动生成包含 JavaScript 和 GeoGebraApplet 代码的文档, 通过 JavaScript 与 GeoGebraApplet 的接口实现了几何图形的动态呈现, 具体过程如下. 由于几何描述语言的概念实例允许嵌套 (即层数大于 1), 因此首先需要将嵌套形式的实例转化为一个构型, 转化规则如下: 若一个实例形如 $f(a_1,\cdots,a_n)$, 并且 $a_{i_1},\cdots,a_{i_m}$ $(1 \leqq m \leqq n)$ 为概念的实例, 则生成一个新的构型 configuration$(v_1 := a_{i_1},\cdots,v_m := a_{i_m})$, 其中 v_j $(1 \leqq j \leqq m)$ 都为 GDL 表述中未出现过的不同变量. 这样所有实例都不存在嵌套形式, 并且所有分句都是引用分句或者声明分句. 接下来, 将形如 $V := \mathcal{K}$ 的引用分句转换为 $V = \mathcal{K}$ 或者 $V' = \mathcal{K}$ (当 $\mathcal{K}$ 形如 rotate(A, B, C) 时), 将 $\mathcal{K}$ 映射为 GeoGebra 中的作图指令 (见表 5.7), 再将形如 P::Point 的抽象实例转化为 $P = (a, b)$, 其中 (a, b) 是随机生成的点的坐标. 由于 GeoGebra 的作图过程是按作图指令序列依次执行的, 当前指令中所出现的变量需要在执行此指令之前被构造, 因此将得到的表述按如下规则进行转化得到作图表述 Proposition$(N, T, \mathcal{H}, \mathcal{C})\mapsto$ Diagram$(N, \{\eta_1;\cdots;\eta_k\}, \text{GeoGebra})$ 其中 $\eta_1,\cdots,\eta_k$ 为 $\mathcal{H}$ 和 $\mathcal{C}$ 中的分句经过上述处理得到的排序后的作图指令序列.

表 5.7　**将基本概念的实例映射为 GeoGebra 中的作图指令**

实例	作图指令
point()	(a, b) (随机生成的点的坐标)
pointon(A)	Point$[A]$
circle(A, B, C)	Circle$[A, B, C]$
circle(A, B)	Circle$[A, B]$
center(A)	Center$[A]$
intersection(A, B)	Intersection$[A, B]$
tangentpoint(A, B)	Intersection$[A, B]$
perpendicularline(A, B)	PerpendicularLine$[A, B]$
parallelline(A, B)	Line$[A, B]$
line(A, B)	Line$[A, B]$
halfline(A, B)	Ray$[A, B]$

表 5.7 **将基本概念的实例映射为 GeoGebra 中的作图指令 (续)**

实例	作图指令
midpoint(a)	Midpoint$[a]$
segment(A, B)	Segment$[A, B]$
bisectorline(A, B, C)	AngleBisector$[A, B, C]$
rotate(A, B, C)	Angle$[A, B, C]$
triangle(A, B, C)	Polygon$[A, B, C]$
quadrilateral(A, B, C, D)	Polygon$[A, B, C, D]$
conic(A, B, C, D, E)	Conic$[A, B, C, D.E]$
symmetry(A, B)	Reflect$[A, B]$
2tangentline(A, B)	Tangent$[A, B]$
distance(A, B)	Segment$[A, B]$
angle(A, B, C)	Angle$[A, B, C]$
arc(A, B, C)	CircumcircularArc$[A, B, C]$

目前, 我们为 GeoData 所存储的一部分定理自动构造了相应的动态几何图形, 其余的测试工作仍在进行当中. 图 5.12 展示了自动构造并应用 GeoGebra 绘制教科书中西门松定理的动态几何图形.

5.5 总结

电子几何教科书系统提供了一个整合的环境方便用户来管理和共享教科书包含的知识数据, 交互式地构建动态的几何教科书, 实时地发布多版本的样式化文档. 在构建教科书的过程中, 其结构一致性、完备性以及冗余性可以实时地被系统检测, 从而保证所构建教科书的合理性. 知识对象封装了不同版本的知识数据可以实现不同方面的应用, 例如, 多种自然语言版本内容使得教科书可以在国际化的环境下使用, GDL 表述的命题可以被自动证明, 动态的几何图形可以被自动构造并绘制, 知识对象之间的特定关系可以被自动发现. 读者可访问 `http://geo.cc4cm.org/text/geotext.html` 观看系统的演示视频.

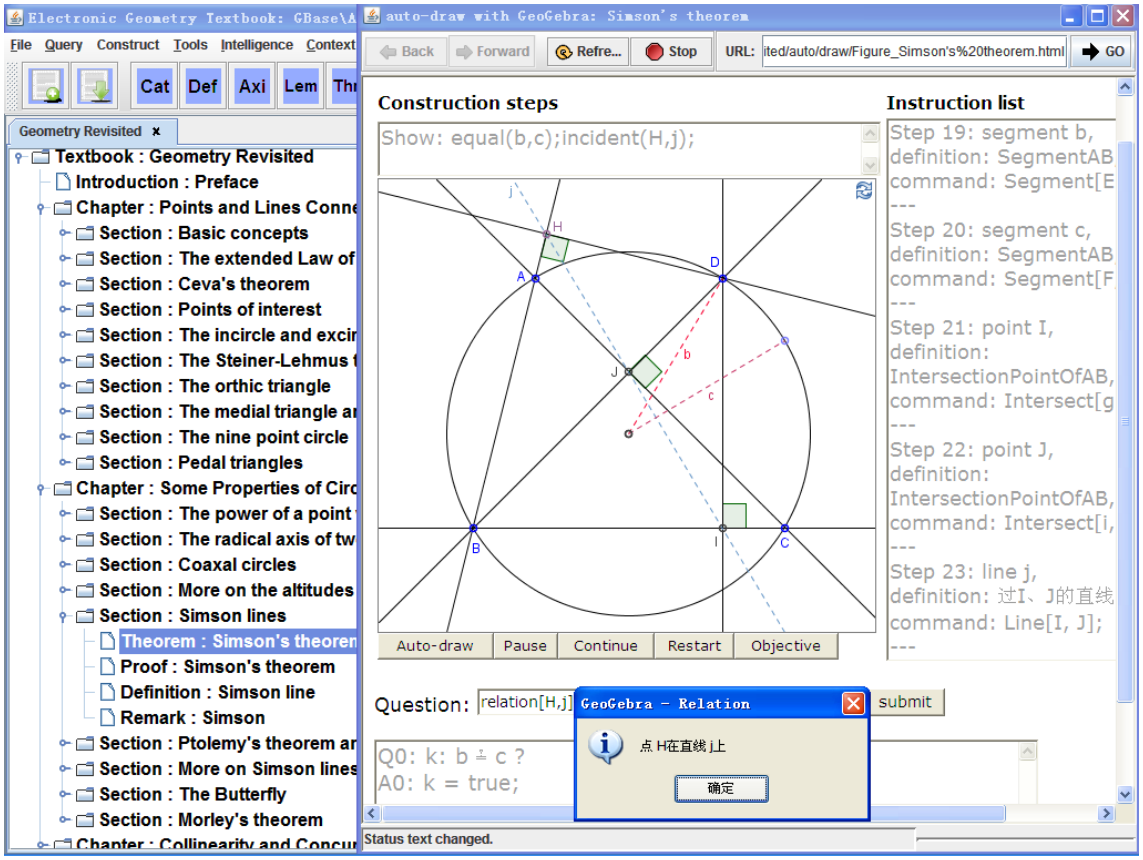

图 5.12 **调用 GeoGebraApplet 自动呈现西门松定理的动态几何图形**

第六章 总结与展望

6.1 总结

本学位论文在数学知识管理研究的大背景下, 将研究对象集中于几何学领域, 通过分析几何知识的范围和区别于其它数学知识的多样性的特点, 提出了一个新的研究方向 —— 几何知识管理, 并根据几何学机械化研究的现状具体阐述了此方向的研究目标、内容和面临的挑战. 我们提出了一套管理和处理几何知识的基本方法和技术, 从几何知识与元知识数据的构建、维护和转化到几何知识对象的自动推理、计算和可视化, 分析讨论了每一步的主要任务、问题及其解决思路.

通过分析传统教科书的静态文档模式的不足, 我们应用所提出的方法和技术, 设计并实施了以动态的、易于维护的、计算机可处理的教科书的形式来管理几何知识的系统 —— 电子几何教科书系统. 我们设计、实现并构建了一个包含近千条定理与定义并具有数据维护、浏览和检索功能的几何知识库. 以此为平台, 教科书的内容可以通过合适的粒度进行存储与组织, 不同版本的知识数据可以得到有效地管理, 从而提高了教科书的创建和使用效率. 我们研究并解决了如何实时地检测教科书的结构一致性、完备性以及冗余性问题, 使得系统可以辅助用户构建符合传统的叙述组织规则, 内容全面且没有重复的教科书.

教科书所包含的知识内容需要通过多种语言来表述以满足不同的应用需求. 我们使用 XML 来标注几何知识的自然语言表述并开发了相应的样式表, 使得教科书文档以传统自然的样式呈现给读者阅读浏览及打印. 为了整合与利用现有的几何软件来处理教科书中的内容并增加教科书的交互性, 我们通过分析几何知识的自然语言表述结构, 研究并设计了易于阅读和使用的几何描述语言, 用来形式化几何概念、分句、构型、定义、命题、问题等. 通过分析并模拟人们在解决问题时所进行的概念替换过程, 我们基于几何描述语言, 提出了使用概念的定义对几何构型、命题、问题等几何表述进行同义转化的自动化方法, 使得转化后的表述可以经过简单的概念映射便可被外部软件工具识别和处理, 从而实现了系统与外部几何软件之间的接口, 使得教科书中的几何命题或证明问题可以被自动证明并且相应的动态几何图形可以被自动构造并绘制. 另外, 基于几何描述语言表述, 我们研究并实现了自动发现知识对象之间语境关系以及定义对象之间继承

关系的方法, 从而辅助获取元知识数据.

综上所述，本文研究工作的成果及创新点总结如下:

- 开辟了几何知识管理这个研究方向, 提出了一套获取、表示、封装、分类、组织、维护、呈现、转化、处理几何知识的管理方法和技术, 为此方向的研究制定了初步的理论框架;
- 设计构建的几何知识库可以服务于自动推理与计算、自动作图、自动翻译、同义转化、多版本文档生成等应用, 为几何软件的开发提供精细的数据规格, 为几何研究提供可测试的标准数据, 也为几何教学提供丰富的学习资源;
- 设计实施的电子几何教科书系统展示了一种动态的、有效的、智能化教科书的新模式, 弥补了传统教科书在创建、共享、交互、更新、演化以及版本维护等方面的不足. 此系统的研制与开发将会对电子化知识管理、文档编写、文本处理、书籍出版以及几何教学等方面的基础和应用研究产生深远影响;
- 设计的几何描述语言易于使用和阅读, 实现的几何表述同义转化的相关方法为几何知识的智能管理, 几何教科书等文献资源中定理的自动证明及其动态图形的自动生成等, 提供了有力的工具.

电子几何教科书系统的设计、开发并最终应用到实际需要一个长期的过程, 我们的研究刚刚起步, 系统仍处于研究开发阶段而并没有被发行. 因此,目前系统的测试与评估工作仅由作者以及相关开发人员进行. 下一步将会请相关教师和学生在实际中进行使用测试，并根据反馈评估意见进一步完善系统. 下面我们针对其中未完善的部分以及待解决的问题对未来的发展和工作做一个展望, 并期望可以为几何知识管理乃至数学知识管理的研究提供一些新的思路和途径.

6.2　几何知识表述语言及其应用

几何知识具有多种类型不同形式, 如陈述型知识中定义的几何形式、定理的代数形式等, 过程化知识中的转化规则、算法等. 为了使几何知识可以全面地被计算机理解和处理, 我们需要统一的形式语言来表述各种类型和形式的几何知识. 目前几何描述语言的表达能力还不足够强大, 不能表述定理的证明、问题的解答等陈述型几何知识, 因而需要被扩展. 另外, 一些过程化几何知识, 如算法、机械化证明方法等的形式化表述方法也有待研究. 基于这种形式语言, 我们需要研究相应的算法, 开发有效的工具实现如下自

动化功能, 从而为几何知识的管理提供多方面全方位的支持.

6.2.1 几何知识不同表述之间的转化

如第 2.1 节所述, 陈述型几何知识的表述涵盖三种不同形式. 例如, 使用自然语言的几何表述可以供人阅读浏览, 代数表述可以为几何计算和推理提供服务, 作图表述可以被相应的动态几何软件处理来直观地呈现几何构型. 实现这些表述之间的相互转化将有利于高效地管理几何知识, 因为这样我们只需要使用一种标准的形式语言来表述几何知识, 而相应的其它表述可以通过自动转化得到. 这也将为几何知识库的构建以及电子几何教科书的创建提供更简洁方便的操作模式.

如图 6.1 所示, 通过应用第 4.3.3 节中所形式化的翻译规则、代数化规则、图形化规则等规则型知识可以在概念的层次上将形式化的几何表述转化为相应的自然语言表述、代数表述、作图表述. 而这些转化的逆过程也是值得研究的. 例如, 将自然语言表述转化为形式化的几何表述可以使用户摆脱形式语法的约束而使得几何知识数据的创建更自然方便; 将代数表述转化为形式化的几何表述可以为坐标代数法增添几何性从而使得定理的正确性获得明确的几何意义 (因为这些方法会自动生成大量的代数语言表达的非退化条件); 将作图表述转化为形式化的几何表述可以使用户应用动态几何软件作为交互界面从而摆脱文本的输入. 当然, 这些逆转化过程的实现存在一定的难度, 只能在一定范围内进行.

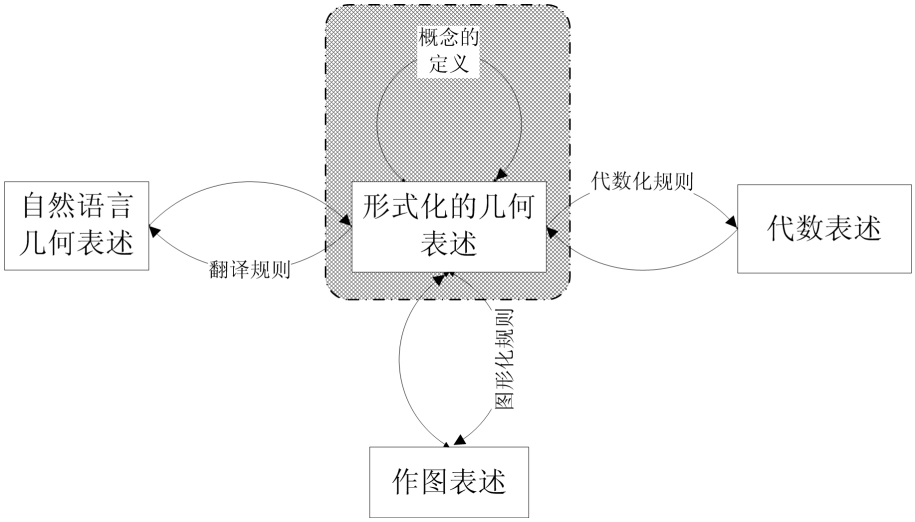

图 6.1 **陈述型知识表述之间的转化**

另一方面, 新的概念可以不断地被引入, 其定义也将成为新的知识. 由于规则型知识

都是针对概念的实例引入的, 因此为了使得转化规则的集合保持最小, 即当引入新的概念时, 相应的代数化规则和图形化规则不必引入, 我们需要应用概念的形式化定义对形式化的几何表述进行同义转化, 使得结果表述中只含有基本概念集中概念的实例 (即图 6.1 中阴影部分的转化), 从而使得已有的转化规则可以被应用. 我们所设计的几何描述语言可以作为形式化的几何表述语言, 并且同义转化的方法在第 4.4.2 节已经详细地讨论并实现了. 其它根据相应的转化规则进行的转化过程类似于对 GDL 表述进行的同义转化 (区别在于后者使用的是概念的定义作为转化规则), 可以参考文中的相应方法.

以上所提出的各种转化依赖于不同表述之间存在的对应关系, 最主要的对应关系在于概念层次上的对应. 例如, 几何表述中的一个圆对应代数表述中的一个二次方程, 又对应作图表述中的一个作图指令. 而对于更高层次的对应, 例如定义的结构 (...称为...), 证明的结构 (因为..., 所以...), 定理的结构 (如果..., 那么...) 等, 可以根据应用需求在具体的程序中 (或者依然通过形式地定义相应规则的手段) 实现.

在形式化的几何表述自动转化的基础上, 我们可以研究新的几何文档创建与呈现模式. 由于这些表述可以部分地 (例如概念的实例) 进行自动转化, 因此可以采取自然语言与形式语言混合使用的半形式化方式来创建几何文档, 这样的文档既可以保持自然语言表述的易读性, 同时其中的形式语言表述部分又可以自动转化为其它表述来动态地诠释其含义. 因此在呈现这样的文档时, 首先需要进行预处理, 即将形式语言表述部分自动转化为自然语言表述、代数表述以及作图表述, 然后在呈现的过程中, 用户可以交互式地选择查看所需要的表述. 例如在语句“给定 $circle(O,r)$”中, $circle(O,r)$ 是形式语言表述的部分, 它被嵌入到自然语言的表述当中, 可以被自动转化为如“以 O 为圆心 r 为半径的圆”、“the circle with O as its center and r as its radius”的自然语言表述, 或者“$(x-a)^2+(y-b)^2=r^2$”的代数方程, 或者“$Circle[O,r]$”的作图指令并通过调用相应的动态几何软件自动绘制这个圆的动态几何图形. 这样的文档可以增加阅读的直观体验, 方便用户理解文档的内容.

6.2.2　几何证明的验证与动态几何图形的自动生成

对于几何命题和 (证明) 问题, 我们可以使用几何描述语言对其进行完全地形式化, 并通过同义转化方法对其进行自动证明 (见第 5.4.4 节), 从而辅助判断其正确性. 然而从另一个角度来看, 给定一个命题的证明, 通过判断证明的正确性也可以间接地判断此命题的正确性, 并且这种对证明的验证在实际应用当中有着重要的意义, 因为证明刻画了

几何知识之间的逻辑关系, 是几何知识的元知识获取的主要来源. 证明的正确性检测目前还只能由人来完成, 在审阅数学文档或论文中的数学理论时, 最主要的任务就是验证证明. 如果可以使用计算机来辅助检测证明的正确性将会对数学的发展和知识的传播起到深远的影响. 对证明的验证依赖于证明所使用的表述语言. [89,98] 展示了使用证明助手 Coq, 在对几何公理和定义进行严格形式化的基础上, 通过应用系统所提供的逻辑推导规则来构造形式证明的方法. 由于证明的每一步都是经过系统验证的, 因此若能够最终构造成功, 那么所得证明必然是正确的. 虽然这种形式证明是经过验证的, 但是证明过程比较繁琐, 并不能直观地反应证明的几何意义. 因此需要研究并设计自然方便地表述几何证明的形式语言, 并利用现有的或者开发新的工具来检测其正确性.

另一方面, 形式化的几何构型、命题和问题经过同义转化后可以通过动态几何软件来自动呈现相应的动态几何图形 (见第 5.4.5 节), 然而这些软件大部分都是按照作图指令的次序一步一步地绘制几何图形, 每一步的执行都依赖于已构造 (或存在) 的几何对象, 因此我们在规约时所使用的概念定义都是构造型的, 所得到的同义转化结果也是构造型的几何构型. 一个值得研究的问题是给定一个约束型的几何构型, 如何自动生成满足其中约束条件的动态几何图形而不必考虑构造图形时必须面对的作图次序问题. 这涉及到几何约束求解技术, 在 [63,126] 中有相关方法的讨论, 但仍存在很多问题有待解决.

6.2.3 元知识的自动发现

目前, 几何知识的元知识数据大部分都是人为获取的. 在第 4.4.3 节, 我们提出了自动发现知识对象之间语境关系和定义对象之间继承关系的方法, 然而知识对象之间其它类型的关系并没有被深入研究和自动发现. 因此, 如何利用几何知识库所存储的大量几何知识数据, 挖掘知识对象之间的关系以及分类体系是一个值得研究的问题. 例如, 从证明的形式表述中自动抽取出所使用的定理或引理等知识对象, 它们将与此证明对象构成蕴含 (Implication) 关系 (见第 3.2.3 节). 关系的自动发现方法依赖于几何知识的形式化表述, 并且其研究也依赖于人们对元知识的认识和理解程度.

6.2.4 几何知识的搜索

在第 3.3.2 节, 我们给出了利用元知识数据来进行宏观检索的方法. 由于元知识数据的结构比较简单, 而且检索是在知识对象的层次上进行的, 因此这种检索比较初等. 如何通过几何表述本身来进行检索以获得所需要的知识对象是一个值得研究的问题. 几何知

识都是基于几何构型的, 因此如何匹配几何构型将是研究的重点. 首先, 由于应用的概念不同, 相同几何构型的表述将会不同, 因此仅匹配关键字并不能满足要求; 另一方面, 由于构型可以是构造型的, 也可以是约束型的, 其表述的不同类型会给匹配带来障碍, 因此, 需要制定一个规范来约束几何构型的表述. 几何描述语言作为用于几何表述的形式语言, 为几何构型的规范化提供了以下可应用的方法和条件. 一方面, 几何构型可以在合适的脚本下进行同义转化, 使得结果仅应用脚本的基本概念集中的概念. 由于基本概念不能再由其它概念定义, 因此避免了转化结果在概念应用上的不同; 另一方面, 脚本可以由约束型定义或者构造型定义构成, 这样便消除了转化结果在表述类型上的障碍. 因此, 几何构型表述的同义转化结果可以作为其匹配的标准形式. 以此为基础, 应用信息技术中的方法进行匹配操作, 可以为几何知识的搜索提供新的途径和有力工具.

6.3　教学上的应用

目前, 电子几何教科书系统能够辅助人们有效地创建、维护、共享、定制、浏览不同版本的传统意义上的几何教科书, 虽然具有自动证明几何定理与自动呈现动态几何图形的功能, 但是从其在教学上的使用来看, 仍然缺少了与学习者之间的交互功能. 因此, 需要增加其交互性, 使其成为能够辅助学生学习几何知识的智能辅导系统.

6.3.1　交互式几何练习

学生学习理解并掌握知识的必不可少的过程便是做练习. 随着智能辅导系统研究的发展, 出现了很多交互式地辅助学生做练习的工具. 这些工具可以评估学生对问题的回答, 根据一定的教学策略自动生成合适的反馈或者提示信息, 从而一步一步地引导学生完成练习. 这样的交互式练习需要使用标准的格式来表示其中所包含的多种类型不同层次的反馈和提示信息以及交互策略 [53,61], 还需要计算机代数系统以及领域推理工具的辅助来评估回答步骤中的计算和推理过程. 目前这种交互式练习工具在代数、微积分等数学领域的网络教学中应用比较广泛. 在几何领域, 一些功能强大的动态几何软件 (如超级画板) 提供了课件制作与进行交互式几何计算和推理练习的功能, 但这些功能都是在软件内部通过不同的方法和技术实现的, 并不能在不同的软件之间共享和重用. 因此, 需要研究并设计一种统一的格式来表示不同类型的交互式几何练习, 并通过与几何软件的交互来评估和诊断回答, 进而生成反馈和提示信息. 由于几何练习的解答通常需要图形的辅助, 因此如何能够独立于图形来规范几何问题的解答步骤以及交互策略将是研究的

重点和难点.

6.3.2 自适应教科书的生成

应用电子几何教科书系统所构建的教科书事实上反应了教科书创建者的教学目的, 然而这样的教科书并不一定适合所有学习者各自的动机、水平、能力、特点和兴趣等, 因此需要进一步研究如何根据学习者的个性化信息自动生成合适的教科书以满足他们各自的学习需求, 以便提高教科书的使用和学习者的学习效率. 这涉及到学生建模和自适应课程生成两个相关领域的方法和技术. 系统可以根据学习者详细描述的学习目标自动搜集最适合其目标和能力等参数的知识内容, 并选取合适的练习生成教科书, 在学习者遇到问题时自动诊断并生成提示信息以帮助其克服困难. 这样的系统需要构建于学习者的个性化模型和教学的辅导策略. 教科书知识库中存在着大量的几何知识数据以及有关教学规则的元知识数据, 因此, 需要在此基础上扩展其元知识并制定合理的辅导策略, 从而自适应地生成满足不同学习者需求的教科书.

参考文献

[1] Ab*á*nades M., Botana F., Escribano J., et al. The Intergeo File Format in Progress[A]. Proceedings of the 22nd OpenMath Workshop[C]. Ontario, Canada, 2009:17-30

[2] Abramowitz M. and Stegun I. A. Handbook of Mathematical Functions (with Formulas, Graphs, and Mathematical Tables)[M]. Dover Publications, New York, 1965

[3] ActiveMath, `http://www.activemath.org/`

[4] Algebra Interactive, `http://www.win.tue.nl/~ida/home.html`

[5] Asperti A., Coen C. S., Tassi E., et al. User Interaction with the Matita Proof Assistant[J]. Aspinall D. and Lüth C. (eds.), Special Issue on User Interfaces in Theorem Proving, Journal of Automated Reasoning, 2007, V39(2):109-139

[6] Asperti A., Guidi F., Coen C. S., et al. A Content Based Mathematical Search Engine: Whelp[A]. Filliâtre J., Paulin-Mohring C. and Werner B. (eds.), Proceedings of the TYPES 2004 Workshop (TYPES 2004)[C]. LNCS **3839**, Springer-Verlag, Berlin Heidelberg, 2006:17-32

[7] Asperti A., Padovani L., Coen C. S., et al. Content-centric Logical Environments [R]. Short presentation at the 15th Annual IEEE Symposium on Logic in Computer Science (LICS 2000), Santa Barbara, California, USA, June 2000

[8] Asperti A., Padovani L., Coen C. S., et al. HELM and the Semantic Math-Web[A]. Boulton R. J. and Jackson P. B. (eds.), Proceedings of the 14th International Conference on Theorem Proving in Higher Order Logics (TPHOLs 2001)[C]. LNCS **2152**, Springer-Verlag, Berlin Heidelberg, 2001:59-74

[9] Asperti A., Padovani L., Coen C. S., et al. Mathematical Knowledge Management in HELM[J]. Special Issue on Mathematical Knowledge Management, Annals of Mathematics and Artificial Intelligence, 2003, V38(1-3):27-46

[10] Asperti A. and Wegner B. MOWGLI – A New Approach for the Content Description in Digital Documents[A]. Proceedings of the 9th International

Conference on Electronic Resources and the Social Role of Libraries in the Future[C], 2002, V1, Autonomous Republic of Crimea, Ukraine

[11] Bancerek G. and Rudnicki P. Information Retrieval in MML[A]. Asperti A., Buchberger B. and Davenport J. H. (eds.), Proceedings of the 2nd International Conference on Mathematical Knowledge Management (MKM 2003)[C]. LNCS **2594**, Springer-Verlag, Berlin Heidelberg, 2003:119-132

[12] Brown C. E. Verifying and Invalidating Textbook Proofs Using Scunak[A]. Borwein J. M. and Farmer W. M. (eds.), Proceedings of the 5th International Conference on Mathematical Knowledge Management (MKM 2006)[C]. LNAI **4108**, Springer-Verlag, Berlin Heidelberg, 2006:110-123

[13] Bruijn N. G. The Mathematical Vernacular, A Language for Mathematics with Typed Sets[A]. Nederpelt R. P., Geuvers J. H. and Vrijer R. C. (eds.), Selected Papers on Automath[M]. Elsevier, North Holland, 1994:865-935

[14] Buchberger B. Algorithm Retrieval: Concept Clarification and Case Study in Theorema[R]. SFB Report no. 2003-44, Johannes Kepler University, Linz, Austria, October 2003

[15] Buchberger B. Mathematica: Doing Mathematics by Computer?[R]. RISC Report no. 93-50, invited talk at the International Symposium on Design and Implementation of Symbolic Computation Systems (DISCO 1993), Gmunden, Austria, September 1993

[16] Buchberger B., Craciun A., Jebelean T., et al. Theorema: Towards Computer-Aided Mathematical Theory Exploration[J]. Journal of Applied Logic, 2006, V4(4):470-504

[17] Buchberger B., Gonnet G. and Hazewinkel M. (eds.), Special Issue on Mathematical Knowledge Management. Annals of Mathematics and Artificial Intelligence, V38, Kluwer Academic Publisher, Dordrecht, 2003

[18] Cabri 3D, `http://www.cabri.com/download-cabri-3d.html`

[19] Carette J. and Farmer W. M. A Review of Mathematical Knowledge Management [A]. Carette J., Dixon L., Coen C. S., et al. (eds.), Proceedings of the 8th

International Conference on Mathematical Knowledge Management (MKM 2009)[C]. LNAI **5625**, Springer-Verlag, Berlin Heidelberg, 2009:233-246

[20] 曹存根, 眭跃飞, 孙瑜, 等. 国家知识基础设施中的数学知识表示[J]. 软件学报, 2006, V17(8):1731-1742

[21] Chen X. Electronic Geometry Textbook: A Geometric Textbook Knowledge Management System[A]. Autexier S., Calmet J. and Delahaye D. (eds.), Proceedings of the 9th International Conference on Mathematical Knowledge Management (MKM 2010)[C]. LNAI **6167**, Springer-Verlag, Berlin Heidelberg, 2010:278-292

[22] Chen X. Formal Representation and Automated Transformation of Geometric Statements[A]. Richter-Gebert J. and Schreck P. (eds.), Proceedings of the 8th International Workshop on Automated Deduction in Geometry (ADG 2010)[C]. Technical University Munich, Germany, 2010:1-19

[23] Chen X., Huang Y. and Wang D. On the Design and Implementation of a Geometric knowledge Base[A]. Sturm T. and Zengler C. (eds.), Proceedings of the 7th International Workshop on Automated Deduction in Geometry (ADG 2008)[C]. LNAI **6301**, Springer-Verlag, Berlin Heidelberg, 2011:22-41

[24] Chen X. and Wang D. Towards an Electronic Geometry Textbook[A]. Botana F. and Recio T. (eds.), Post-proceedings of the 6th International Workshop on Automated Deduction in Geometry (ADG 2006)[C]. LNAI **4869**, Springer-Verlag, Berlin Heidelberg, 2007:1-23

[25] Chen X. and Wang D. Management of Geometric Knowledge in Textbooks[J]. Data & Knowledge Engineering (DKE), 2011, in press

[26] Chou S. Mechanical Geometry Theorem Proving[M]. Reidel, Dordrecht, 1988

[27] Chou S. and Gao X. A Survey of Geometric Reasoning Using Algebraic Methods [A]. Kueker D. W. and Smith C. (eds.), Learning and Geometry: Computational Approaches[M], Birkhäuser, Boston, 1996:97-119

[28] Chou S. and Gao X. Automated Reasoning in Geometry[A]. Robinson A. and Voronkov A. (eds.), Handbook of Automated Reasoning (Volume I)[M], Elsevier,

North Holland, 2001:707-749

[29] Chou S., Gao X. and Zhang J. Machine Proofs in Geometry[M]. World Scientific, Singapore, 1994

[30] Chou S., Gao X. and Zhang J. Automated Generation of Readable Proofs with Geometric Invariants I & II[J]. Journal of Automated Reasoning, 1996, V17:325-370

[31] Chou S., Gao X. and Zhang J. A Deductive Database Approach to Automated Geometry Theorem Proving and Discovering[J]. Journal of Automated Reasoning, 1996, V25:219-246

[32] Cinderella, http://cinderella.de/

[33] Colton S., McCasland R., Bundy A., et al. Automated Theory Formation for Tutoring Tasks in Pure Mathematics[A]. Proceedings of the Workshop on the Role of Automated Deduction in Mathematics (RADM 2002)[C]. Copenhagen, Denmark, 2002

[34] Coq Proof Assistant, http://coq.inria.fr/

[35] Coxeter H. S. M. and Greitzer S. L. Geometry Revisited[M]. The Mathematical Association of America, Washington D.C., 1967

[36] Cruz-Filipe L. Constructive Real Analysis: A Type-theoretical Formalization and Applications[D]. Radboud University Nijmegen, the Netherlands, 2004

[37] Daconta M. C., Obrst L. J. and Smith K. T. The Semantic Web: A Guide to the Future of XML, Web Services and Knowledge Management[M]. Wiley Publishing, Indiana, 2003

[38] Davenport J. H. and Kohlhase M. Unifying Math Ontologies: A Tale of Two Standards[A]. Carette J., Dixon L., Coen C. S., et al. (eds.), Proceedings of the 8th International Conference on Mathematical Knowledge Management (MKM 2009)[C]. LNAI **5625**, Springer-Verlag, Berlin Heidelberg, 2009:263-278

[39] Digital Library of Mathematical Functions (DLMF), http://dlmf.nist.gov/

[40] Egido S., Hendriks M., Kreis Y., et al. i2g Common File Format Final Version, http://i2geo.net/files/D3.10-Common-File-Format.pdf

[41] Farmer W. M. MKM: A New Interdisciplinary Field of Research[J]. ACM SIGSAM Bulletin, 2004, V38(2):47-52

[42] Fèvre S. Integration of Reasoning and Algebraic Calculus in Geometry[A]. Wang D. (ed.), Prodeedings of the 1st International Workshop on Automated Deduction in Geometry (ADG 1996)[C]. LNAI **1360**, Springer-Verlag, Berlin, 1998:218-234

[43] Fujimoto M. and Watt S. M. An Interface for Math e-Learning on Pen-Based Mobile Devices[A]. Proceedings of the Workshop on Mathematical User Interfaces (MathUI 2010)[C]. CNAM, Paris, 2010

[44] 高小山, 张景中, 周咸青. 几何专家[M]. 台北: 九章出版社, 1998

[45] GCLC, `http://poincare.matf.bg.ac.rs/~janicic//gclc/`

[46] GeoGebra, `http://www.geogebra.org/cms/`

[47] GeoGebra Applet Methods, `http://www.geogebra.org/en/wiki/index.php/GeoGebra_Applet_Methods`

[48] Geometry Expert, `http://www.mmrc.iss.ac.cn/gex/`

[49] Geometry Explorer, `http://homepages.gac.edu/~hvidsten/explorer/`

[50] Geometry Expressions, `http://geometryexpressions.com/`

[51] GeoProof, `http://home.gna.org/geoproof/`

[52] GeoProver, `http://www.reduce-algebra.com/docs/geoprover.html`

[53] Goguadze G. Representation for Interactive Exercises[A]. Carette J., Dixon L., Coen C. S., et al. (eds.), Proceedings of the 8th International Conference on Mathematical Knowledge Management (MKM 2009)[C]. LNAI **5625**, Springer-Verlag, Berlin Heidelberg, 2009:294-309

[54] Gräbe H. The SymbolicData GEO Records – A Public Repository of Geometry Theorem Proof Schemes[A]. Winkler F. (ed.), Proceedings of the 4th International workshop on Automated Deduction in Geometry (ADG 2002)[C]. LNAI **2930**, Springer-Verlag, Berlin Heidelberg, 2004:67-86

[55] Grabowski A. and Schwarzweller C. On Duplication in Mathematical Repositories[A]. Autexier S., Calmet J., Delahaye D., et al. (eds.), Proceedings of the 9th International Conference on Mathematical Knowledge Management

(MKM 2010)[C]. LNAI **6167**, Springer-Verlag, Berlin Heidelberg, 2010:300-314

[56] Gruber T. Ontology[A]. Liu L. and Özsu M. T. (eds.), Encyclopedia of Database Systems (Springer Reference)[M], Springer-Verlag, 2009:1963-1965

[57] Guidi F. Searching and Retrieving in Content-based Repositories of Formal Mathematical Knowledge[D]. University of Bologna, Italy, 2003

[58] Hadamard J. Lessons in Geometry: I. Plane Geometry[M]. American Mathematical Society, Providence, 2008

[59] Hales T. C. Formal Proof[J]. A Special Issue on Formal Proof, Notices of the American Mathematical Society, 2008, V55:1370-1381

[60] Harrison J. HOL Light: A Tutorial Introduction[A]. Proceedings of the 1st International Conference on Formal Methods in Computer-Aided Design (FMCAD 1996)[C]. LNCS **1166**, Springer-Verlag, Berlin Heidelberg, 1996. `http://www.cl.cam.ac.uk/~jrh13/hol-light/tutorial_220.pdf`, 2006

[61] Heeren B., Jeuring J., Leeuwen A., et al. Specifying Strategies for Exercises[A]. Autexier S., Campbell J., Rubio J., et al. (eds.), Proceedings of the 7th International Conference on Mathematical Knowledge Management (MKM 2008)[C]. LNAI **5144**, Springer-Verlag, Berlin Heidelberg, 2008:430-445

[62] Heras J., Pascual V., Romero A., et al. Integrating Multiple Sources to Answer Questions in Algebraic Topology[A]. Proceedings of 10th International Conference on Artificial Intelligence and Symbolic Computation (AISC 2010)[C]. LNAI **6167**, Springer-Verlag, Berlin Heidelberg, 2010:331-335

[63] Hong H., Li L., Liang T., et al. Solving Dynamic Geometric Constraints Involving Inequalities[A]. Calmet J., Ida T., Wang D. (eds.), Proceedings of the 8th International Conference on Artificial Intelligence and Symbolic Computation (AISC 2006)[C]. LNAI **4120**, Springer-Verlag, Berlin Heidelberg, 2006:181-195

[64] Horn P. and Roozemond D. OpenMath in SCIEnce: SCSCP and POPCORN[A]. Carette J., Dixon L., Coen C. S., et al. (eds.), Proceedings of the 8th International Conference on Mathematical Knowledge Management (MKM 2009)[C]. LNAI **5625**, Springer-Verlag, Berlin Heidelberg, 2009:474-479

[65] INTERGEO Project, http://i2geo.net/xwiki/bin/view/Main/

[66] Isabelle's Logics, www.cl.cam.ac.uk/research/hvg/isabelle/dist/Isabelle2011/doc/logics.pdf

[67] Java Geometry Expert, http://www.cs.wichita.edu/~ye/

[68] JDesktop Integration Components, http://javadesktop.org/articles/jdic/

[69] jEditOQMath, http://www.activemath.org/projects/jEditOQMath/

[70] JSXGraph, http://jsxgraph.uni-bayreuth.de/wp/

[71] Jucovschi C. and Kohlhase M. sTeXIDE: An Integrated Development Environment for sTeX Collections[A]. Autexier S., Calmet J., Delahaye D., et al. (eds.), Proceedings of the 9th International Conference on Mathematical Knowledge Management (MKM 2010)[C]. LNAI **6167**, Springer-Verlag, Berlin Heidelberg, 2010:336-344

[72] Kamareddine F., Maarek M., Retel K., et al. Narrative Structure of Mathematical Texts[A]. Kauers M., Kerber M., Miner R., et al. (eds.), Proceedings of the 6th International Conference on Mathematical Knowledge Management (MKM 2007)[C]. LNAI **4573**, Springer-Verlag, Berlin Heidelberg, 2007:296-312

[73] Kapur D. and Wan H. K. Refutational Proofs of Geometry Theorems via Characteristic Set Computation[A]. Proceedings of the International Symposium on Symbolic and Algebraic Computation (ISSAC 1990)[C]. American Mathematical Society, Providence, 1990:277-284

[74] Kerber M. (ed.), Special issue on Management of Mathematical Knowledge. Mathematics in Computer Science, Birkhäuser, Basel, 2008, V2(2):193-398

[75] Kohlhase M. OMDoc – An Open Markup Format for Mathematical Documents [version 1.2]: Foreword by Alan Bundy[M]. LNAI **4180**, Springer-Verlag, Berlin Heidelberg, 2006

[76] Kohlhase M., Anca S., Jucovschi C., et al. MathWebSearch 0.4: A Semantic Search Engine for Mathematics[A]. http://mathweb.org/projects/mws/pubs/mkm08.pdf, 2008

[77] Kohlhase M. and Franke A. MBase: Representing Knowledge and Context for the Integration of Mathematical Software Systems[J]. Journal of Symbolic Computation, 2001, V23(4):365-402

[78] Kohlhase M. and Sucan I. A Search Engine for Mathematical Formulae[A]. Ida T., Calmet J. and Wang D. (eds.), Proceedings of the 8th International Conference on Artificial Intelligence and Symbolic Computation (AISC 2006)[C]. LNAI **4120**, Springer-Verlag, Berlin Heidelberg, 2006:241-253

[79] Kutzler B. and Stifter S. On the Application of Buchberger's Algorithm to Automated Geometry Theorem Proving[J]. Journal of Symbolic Computation, 1986, V2:389-397

[80] Lange C. Integrated Semantic Web Collaboration on Semiformal Mathematical Knowledge[D]. Jacobs University, Germany, 2010

[81] LaTeXMathML: Translating LaTeX Math Notation Dynamically to Presentation MathML,
`http://www.maths.nottingham.ac.uk/personal/drw/lm.html`

[82] LeActiveMath, `http://www.leactivemath.org/`

[83] Liang T. and Wang D. Towards a Geometric-Object-Oriented Language[A]. Hong H. and Wang D. (eds.), Proceedings of the 5th International Workshop on Automated Deduction in Geometry (ADG 2004)[C]. LNAI **3763**, Springer-Verlag, Berlin Heidelberg, 2006:130-155

[84] Libbrecht P. Notations Around the World: Census and Exploitation[A]. Autexier S., Calmet J. and Delahaye D. (eds.), Proceedings of the 9th International Conference on Mathematical Knowledge Management (MKM 2010)[C]. LNAI **6167**, Springer-Verlag, Berlin Heidelberg, 2010:398-410

[85] Libbrecht P., Desmoulins C., Mercat Ch., et al. Cross-Curriculum Search for Intergeo[A]. Autexier S., Campbell J., Rubio J., et al. (eds.), Proceedings of the 7th International Conference on Mathematical Knowledge Management (MKM 2008)[C]. LNAI **5144**, Springer-Verlag, Berlin Heidelberg, 2008:520-535

[86] Libbrecht P. and Melis E. Methods for Access and Retrieval of Mathematical

Content in ActiveMath[A]. Iglesias A. and Takayama N. (eds.), Proceedings of the 2nd International Congress on Mathematical Software (ICMS 2006)[C]. LNCS **4151**, Springer-Verlag, Berlin Heidelberg, 2006:331-342

[87] List of Interactive Geometry Software, http://en.wikipedia.org/wiki/Interactive_geometry_software

[88] Lozier D. W. NIST Digital Library of Mathematical Functions[J]. Annals of Mathematics and Artificial Intelligence, 2003, V38(1-3):105-119

[89] Magaud N., Narboux J. and Schreck P. Formalizing Projective Plane Geometry in Coq[A]. Sturm T. and Zengler C. (eds.), Proceedings of the 7th International Workshop on Automated Deduction in Geometry (ADG 2008)[C]. LNAI **6301**, Springer-Verlag, Berlin Heidelberg, 2011:141-162

[90] MathDox, http://www.mathdox.org/

[91] MathDox Formula Editor, http://www.mathdox.org/formulaeditor/

[92] Mathematics On the Web: Get it by Logic and Interfaces, http://mowgli.cs.unibo.it/

[93] MathML, http://www.w3.org/Math/

[94] MathPlayer, http://www.dessci.com/en/products/mathplayer/

[95] McCasland R. and Bundy A. MATHsAiD: A Mathematical Theorem Discovery Tool[A]. Jebelean T. and Negru V. (eds.), Proceedings of the 7th International Workshop on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC 2006)[C]. IEEE Computer Society Press, 2006:17-22

[96] Melis E., Büdenbender J., Goguadze G., et al. Knowledge Representation and Management in ActiveMath[J]. Special Issue on Mathematical Knowledge Management, Annals of Mathematics and Artificial Intelligence, 2003, V38(1-3):47-64

[97] Mizar, http://www.mizar.org/

[98] Narboux J. Formalization of Tarski's Geometry in the Coq Proof Assistant. http://dpt-info.u-strasbg.fr/~narboux/tarski.html

[99] Nakagawa K., Nomura A. and Suzuki M. Extraction of Logical Structure from

Articles in Mathematics[A]. Asperti A., Bancerek G. and Trybulec A. (eds.), Proceedings of the 3rd International Conference on Mathematical knowledge management (MKM 2004)[C]. LNCS **3119**, Springer-Verlag, Berlin Heidelberg, 2004:276-289

[100] Nederpelt R. P. Weak Type Theory: A Formal Language for Mathematics[R]. Technische Universiteit Eindhoven, the Netherlands, 2002

[101] Nederpelt R. P., Geuvers J. H. and Vrijer R. C. (eds.), Selected Papers on Automath. Studies in Logic and the Foundations of Mathematics, V133, Elsevier, Amsterdam, 1994

[102] NIST Digital Library of Mathematical Functions, `http://dlmf.nist.gov/`

[103] Ωmega, `http://www.ags.uni-sb.de/~omega/omega/`

[104] OpenMath, `http://www.openmath.org/`

[105] OpenMath Phrasebooks, `http://www.mathdox.org/new-web/projects/phrasebooks.html`

[106] Otter, `http://www-unix.mcs.anl.gov/AR/otter/`

[107] Piroi F. and Buchberger B. Label Management in Mathematical Theories[R]. Technical Report no. 2004-16, Johann Radon Institute for Computational and Applied Mathematics (RICAM), Linz, Austria, November 2004

[108] Prefuse, `http://prefuse.org/`

[109] Quaresma P. and Janičić P. GeoThms — Geometry Framework[R]. Technical Report 2006/002, Centre for Informatics and Systems, University of Coimbra, Portugal, 2006

[110] Rabe F. and Kohlhase M. An Exchange Format for Modular Knowledge[A]. Proceedings of the LPAR Workshops: Knowledge Exchange: Automated Provers and Proof Assistants, and the 7th International Workshop on the Implementation of Logics[C]. CEUR Workshop Proceedings 418, 2008:50-68

[111] Resource Description Framework, `http://www.w3.org/RDF/`.

[112] Richter-Gebert J. Mechanical Theorem Proving in Projective Geometry[J]. Annals of Mathematics and Artificial Intelligence, 1995, V13:139-172

[113] Rosenkranz C. Retrieval and Structuring of Large Mathematical Knowledge Bases in Theorema[D]. Research Institute for Symbolic Computation (RISC), Johannes Kepler University, Linz, Austria, 2009

[114] Rudnicki P. and Trybulec A. Mathematical Knowledge Management in Mizar[A]. Buchberger B. and Caprotti O. (eds.), Electronic Proceedings of the 1st International Workshop on Mathematical Knowledge Management (MKM 2001)[C]. Research Institute of Symbolic Computation (RISC), Johannes Kepler University, Linz, Austria, 2001

[115] SAXON: The XSLT and XQuery Processor, http://saxon.sourceforge.net/

[116] Su W., Wang P, Li L., et al. MathEdit: A Browser-based Visual Mathematics Expression Editor[A]. Proceedings of the 11th Asian Technology Conference in Mathematics (ATCM 2006)[C]. The Hong Kong Polytechnic University, China, 2006

[117] Sutcliffe G., Zimmer J., and Schulz S. TSTP Data — Exchange Formats for Automated Theorem Proving Tools[A]. Distributed Constraint Problem Solving and Reasoning in Multi-Agent Systems[M]. IOS Press, Amsterdam, 2004:201-215

[118] TGTP — Thousands of Geometric problems for geometric Theorem Provers, http://hilbert.mat.uc.pt/TGTP/

[119] The Stylesheet Converts OpenMath Objects to MathML, https://mathdox.org/svn/repos/public/leam2mapleta/openmath2mathml.xsl

[120] Theorema, http://www.risc.jku.at/research/theorema/

[121] Trgalova J., Kortenkamp U., Jahn A., et al. I2geo.net, A Platform for Sharing Dynamic Geometry Resources all over Europe[A]. Poster at the CERME7 Conference, Rzeszow, Poland, 2011. http://i2geo.net/xwiki/bin/download/Main/Proceedings/CERME7poster.pdf

[122] Ullrich C. Description of an Instructional Ontology and its Application in Web Services for Education[A]. Proceedings of Workshop on Applications of Semantic Web Technologies for E-learning (SW-EL'04)[C]. Hiroshima, Japan, 2004:17-23

[123] Ullrich C. Pedagogically Founded Courseware Generation for Web-Based Learning — An HTN-Planning-Based Approach Implemented in PAIGOS[M]. LNAI **5260**, Springer-Verlag, Berlin Heidelberg, 2008

[124] Wang D. Elimination Procedures for Mechanical Theorem Proving in Geometry[J]. Annals of Mathematics and Artificial Intelligence, 1995, V13:1-24

[125] Wang D. Geometry Machines: From AI to SMC[A]. Calmet J.,Campbell J. A., Pfalzgraf J. (eds.), Proceedings of the 3rd International Conference on Artificial Intelligenceand Symbolic Mathematical Computation (AISMC 1996)[C]. LNCS **1138**, Springer-Verlag, Berlin Heidelberg, 1996:213-239

[126] Wang D. Automated Generation of Diagrams with Maple and Java[A]. Joswig M. and Takayama N. (eds.), Algebra, Geometry, and Software Systems[M]. Springer-Verlag, Berlin Heidelberg, 2003:277-287

[127] Wang D. GEOTHER 1.1: Handling and Proving Geometric Theorems Automatically[A]. Winkler F. (ed.), Proceedings of the 4th international workshop on Automated Deduction in Geometry (ADG 2002)[C]. LNAI **2930**, Springer-Verlag, Berlin Heidelberg, 2004:194-215

[128] Wang D. Formalization and Specification of Geometric Knowledge Objects[A]. Hu Z. and Zhang J. (eds.), Proceedings of the 6th Asian Workshop on Foundations of Software (AWFS 2009)[C]. National Institute of Informatics, Tokyo, Japan, 2009:86-98

[129] Wang D. Basic Elements of Computer Geometry[A]. Bouhoula A. and Ida T. (eds.), Proceedings of the Tunisia-Japan Workshop on Symbolic Computation in Software Science (SCSS 2009)[C]. Gammarth, Tunisia, 2009:2-12

[130] Wang D. and Gao X. Geometry Theorems Proved Mechanically Using Wu's Method — Part on Euclidean geometry[J]. Math Mechanical Research, 1987, Preprints, V2:75-106

[131] 王东明, 胡森. 构造型几何定理及其机器证明系统[J]. 系统科学与数学, 1987, V7(2):163-172

[132] Wiedijk F. Mizar: An Impression[Z]. `http://www.mizar.org/project/`

mizarintro.pdf

[133] Wilson S. and Fleuriot J. D. Geometry Explorer: Combining Dynamic Geometry, Automated Geometry Theorem Proving and Diagrammatic Proofs[A]. Proceedings of the 12th Workshop on Automated Reasoning (ARW 2005)[C]. Edinburgh, Scotland, 2005

[134] WolframMathWorld: The Web's Most Extensive Mathematics Resource, http://mathworld.wolfram.com/topics/Geometry.html

[135] Wortel TU/e, http://wortel.tue.nl/

[136] 吴文俊. 初等几何判定问题与机械化证明[J]. 中国科学, 1977, 20:507-516

[137] Wu W.-T. Some Recent Advances in Mechanical Theorem-proving of Geometries[A]. Bledsoe W. W. and Loveland D. W. (eds.), Automated Theorem Proving: After 25 Years[M]. American Mathematical Society, Providence, 1984:235-241

[138] 吴文俊. 几何定理机器证明的基本原理 (初等几何部分)[M]. 北京: 科学出版社, 1984. 英译本: 金小凡, 王东明译. Mechanical Theorem Proving in Geometries: Basic Principles. Springer-Verlag, Wien New York, 1994

[139] Yang L., Hou X. and Xia B. A Complete Algorithm for Automated Discovering of a Class of Inequality-type Theorems[J]. Science China Information Sciences (Science in China Series F), 2001, V44:33-49

[140] Yang L., Zhang J. and Hou X. An Efficient Decomposition Algorithm for Geometry Theorem Proving without Factorization[A]. Shi H. and Kobayashi H. (eds.), Proceedinds of the 1st Asian Symposium on Computer Mathematics (ASCM 1995)[C], Beijing, China, 1995:33-41

[141] Youssef A. Methods of Relevance Ranking and Hit-content Generation in Math Search[A]. Kauers M., Kerber M., Miner R., et al. (eds.), Proceedings of the 6th International Conference on Mathematical Knowledge Management (MKM 2007)[C]. LNAI **4573**, Springer-Verlag, Berlin Heidelberg, 2007:393-406

[142] 曾庆田, 曹存根, 眭跃飞, 等. 基于本体的数学知识获取与知识继承机制研究[J]. 微电子学与计算机, 2003, V20(9):19-27

[143] Zhang J., Chou S. and Gao X. Automated Production of Traditional Proofs for Theorems in Euclidean Geometry I[J]. Annals of Mathematics and Artificial Intelligence, 1995, V13:109-137

[144] 张景中. 超级画板自由行[M]. 北京: 科学出版社, 2006

附　录

A. 几何样例

西门松定理. 一点到一三角形三边的垂足共线当且仅当这点在这个三角形的外接圆上. 这时经过三个垂足的直线称为西门松线.

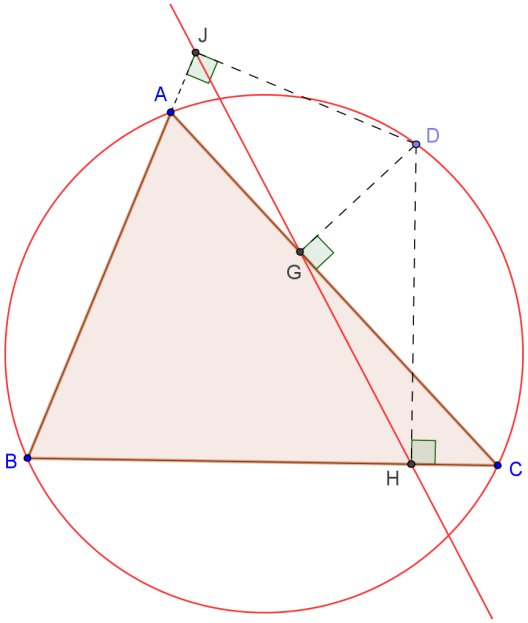

图 1 **西门松定理**

重心定理. 任一三角形的三条中线共点.

帕斯卡定理的逆定理. AB 与 DE 交于 P, BC 与 EF 交于 Q, CD 与 AF 交于 S, P、Q、S 共线, 则 A、B、C、D、E、F 为同一二次曲线上的六点.

毕达哥拉斯定理. 给定一个直角三角形, 则该直角三角形斜边的平方等于同一直角三角形两直角边平方的和.

垂线段定理. 若过直线外一点作一条垂线段和几条斜线段, 则垂线段比任何斜线段都短.

三角形的调和性质. 三角形角平分线被过其它两个角的顶点向该角平分所引的垂线调和分割.

等腰三角形底边中线的性质. 等腰三角形底边的中线也是底边的高线和顶角平分线.

等腰三角形的性质. 在一个等腰三角形中, 底边上一点到两斜边的距离和是常数.

平行直线的性质. 垂直于两条平行直线中一条的直线必与另外一条垂直.

相似三角形的判定定理. 如果一个三角形的两个角与另一个三角形的两个角对应相等, 那么这两个三角形相似.

全等三角形的判定定理. 如果一个三角形的三条边与另一个三角形的三条边对应相等, 那么这两个三角形全等.

帕普斯定理. 设 A、B、C 是一条直线上的三个点, D、E、F 是另一条直线上的三个点, 如果直线 AE、AF、BF 分别与 DB、DC、EC 相交, 则这三个交点 P、Q、R 共线.

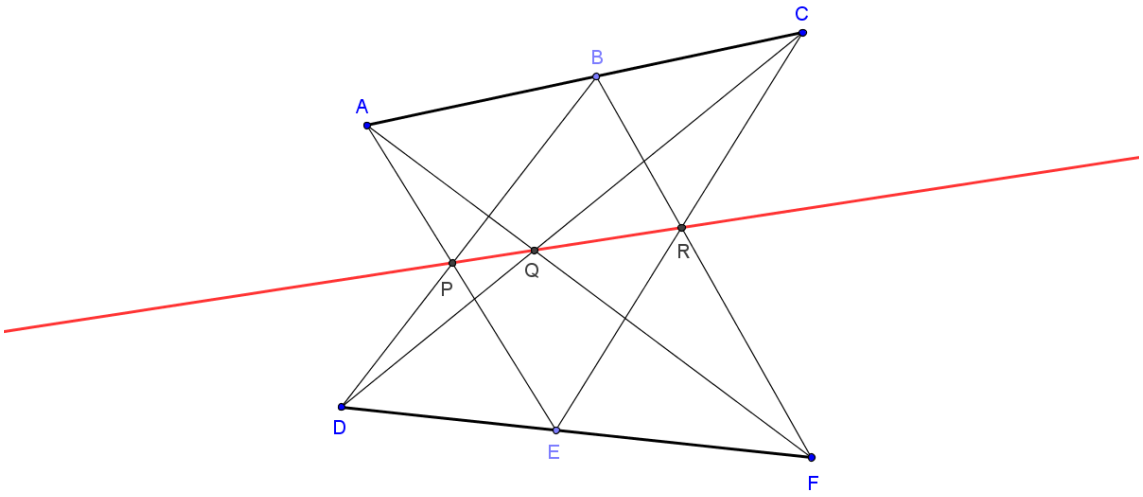

图 2 **帕普斯定理**

三角形的共轭性质. 如果 A_1、B_1、C_1 分别是三角形 ABC 三边中点, 那么 A_1A 是 A_1C 相对于 A_1B_1、A_1C_1 的调和共轭.

圆过点的性质. 给定平面上一个圆和两个点 A、B, 过 A 作圆的割线 AMN 交圆于 M、N , 则过 M、N、B 的圆必过 A 点.

费尔巴哈定理. 三角形的九点圆与内切圆内切, 三角形的九点圆与旁切圆 (三个) 外切.

Nehring 定理. AA_1、BB_1、CC_1 是三角形 ABC 三条共点的赛瓦线. X_1 在 BC 上, X_2 是 X_1B_1 与 BA 的交点, X_3 是 X_2A_1 与 AC 的交点, X_4 是 X_3C_1 与 CB 的交点, X_5 是 X_4B_1 与 BA 的交点, X_6 是 X_5A_1 与 AC 的交点, X_7 是 X_6C_1 与 CB 的交点, 则 X_7 与 X_1 重合.

西门松对偶定理. 连接三角形 ABC 的三个顶点和其内接圆的一条切线与过圆心平行于三角形外角平分线的直线的交点的三条直线共点.

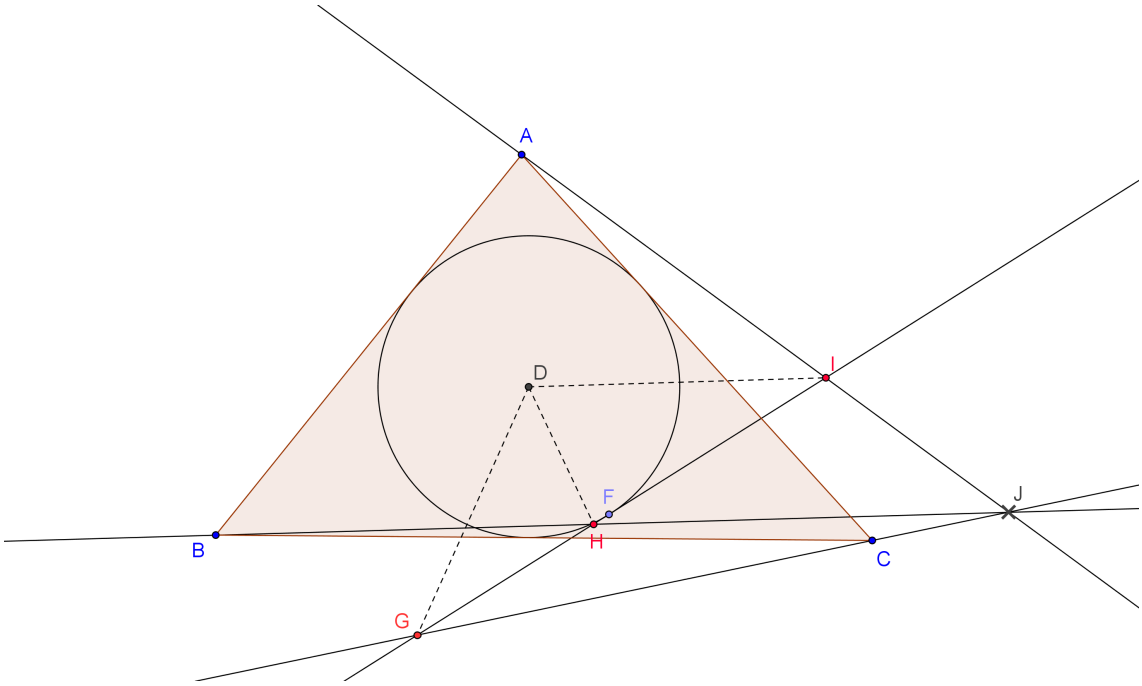

图 3 **西门松对偶定理**

B. 知识类及其数据元素

下表通过在相应的单元格中标记 √ 指明了每个知识类 (表中的行) 包含哪些数据元素 (表中的列). 例如, 公理类 (Axiom) 包含数据元素 knowledgeName、formalRepresentation、naturalRepresentation、diagramInstruction、nondegeneracyCondition、figure、keyWords.

	knowledge-Name	vocabulary attributeList	translation-Script	diagram-Script	algebraic-Script	formal-Representation
Definition	√	√	√	√	√	√
Axiom	√					√
Assertion	√					√
Lemma	√					√
Theorem	√					√
Corollary	√					√
Conjecture	√					√
Proof	√					√
Example	√					√
Exercise	√					√
Solution	√					√
Algorithm	√					√
Method	√					
Introduction	√					
Remark	√					

	natural-Representation	algebraic-Representation	diagram-Instruction	nondegeneracy-Condition	figure	keyWords
Definition	√		√	√	√	√
Axiom	√		√	√	√	√
Assertion	√	√	√	√	√	√
Lemma	√	√	√	√	√	√
Theorem	√	√	√	√	√	√
Corollary	√	√	√	√	√	√
Conjecture	√	√	√	√	√	√
Proof	√				√	√
Example	√	√	√	√	√	√
Exercise	√	√	√	√	√	√
Solution	√				√	√
Algorithm	√					√
Method	√					√
Introduction	√					√
Remark	√					√

C. 电子几何教科书系统使用的 XML 标签

元素	属性			内容
page	language			category
category	name	role	layer	category
categoryNode	name	role		title para
definition	name			naturalRepresentation formalDefinition figure diagramScript algebraicScript translationScript
axiom	name			naturalRepresentation formalRepresentation nondegeneracyCondition figure diagramInstruction
lemma theorem corollary conjecture	name			naturalRepresentation formalRepresentation nondegeneracyCondition algebraicRepresentation diagramInstruction figure
proof	name			figure para
introduction remark	name			title para
err	name			
naturalRepresentation				link para
diagramScript	name			command
command	rank			
figure	name			

元素	属性	内容
	src	
algebraicScript translationScript	name	
diagramInstruction	name	command
algebraicRepresentation	name	coordinatize hypothesis conclusion
title		
coordinatize hypothesis conclusion		equation OpenMath
equation	style:“center” style:“left” style:“right”	OpenMath
link	type:“figure” type:“dynamicFigure” name	
var		

D. 几何描述语言的 XML 表示

元素	属性		内容
	必需	可选	
formaldefinition	author version type		conceptObject return nondegeneracyCondition
conceptObject	type	reference	vocabulary argument
vocabulary			
argument			pointer quantity conceptObject list
pointer	type	reference	
return	count		binding list pointer
nondegeneracyCondition			and or not
constraint			and or not
binding			pointer constraint configuration
and or not	cd		conceptObject and or not
quantity	type	reference	
list			binding pointer

元素	属性		内容
	必需	可选	
			conceptObject reference
quantityFunction	name		
formalDefinitions			formaldefinition
formalAssertions			formalRepresentation
formalRepresentation	author name type		assumption objective
assumption configuration			configuration reference conceptObject and not or list declare give
objective			configuration and not or conceptObject
reference			pointer conceptObject
declare			pointer
give			conceptObject

E. 约束型定义脚本

Script(Definition(point(), A::Point, null),

Definition(line(A::Point, B::Point), a::Line, not(is(A, B))),

Definition(segment(A::Point, B::Point), s::Segment, not(is(A, B))),

Definition(length(segment(A::Point, B::Point)), [distance(A, B)], null),

Definition(halfline(A::Point, B::Point), h::Halfline, not(is(A, B))),

Definition(triangle(A::Point, B::Point, C::Point), t::Triangle, not(collinear(A, B, C))),

Definition(quadrilateral(A::Point, B::Point, C::Point, D::Point), a::Quadrilateral, null),

Definition(angle(A::Point, B::Point, C::Point), a::Angle, and(not(is(A, B)), not(is(B, C)),

not(is(A, C)))),

Definition(angle(line(A::Point, B::Point), line(C::Point, D::Point)), [angle(A, intersection(line(A,

B), line(C, D)), C) where intersect(line(A, B), line(C, D))], null),

Definition(conic(A::Point, B::Point, C::Point, D::Point, E::Point), c::Conic, null),

Definition(size(angle(A::Point, B::Point, C::Point)), s::Degree, null),

Definition(circle(O::Point, r::Length), c::Circle, not(equal(r, 0::Length))),

Definition(arc(A::Point, B::Point, C::Point), a::Arc, null),

Definition(perpendicularline(A::Point, l::Line), [m::Line where and(incident(A, m),

perpendicular(m, l))], null),

Definition(parallelline(A::Point, l::Line), [m::Line where and(incident(A, m), parallel(m, l))],

not(incident(A, l))),

Definition(is(A::Point, B::Point), b::Boolean, null),

Definition(is(line(A::Point, B::Point), l::Line), [and(incident(A, l), incident(B, l))], null),

Definition(is(triangle(A::Point, B::Point, C::Point), triangle(D::Point, E::Point, F::Point)), [

and(is(A, D), is(B, E), is (C, F))], null),

Definition(incident(A::Point, o::Circle), [equal(distance(center(o), A), length(radius(o)))], null),

Definition(incident(A::Point, l::Line), b::Boolean, null),

Definition(incident(A::Point, segment(B::Point, C::Point)), [and(incident(A, line(B, C)),

differentside(A, B, C))], null),

Definition(incident(A::Point, line(B::Point, C::Point)), b::Boolean, null),

Definition(incident(A::Point, halfline(B::Point, C::Point)), [and(incident(A, line(B, C)),

sameside(A, C, B))], null),

Definition(incident(A::Point, bisector(B::Point, C::Point, D::Point)), [equal(distance(A, line(B, C)), distance(A, line(C, D)))], null),

Definition(ratio(A::GeometricQuantity, B::GeometricQuantity), a::AlgebraicQuantity, null),

Definition(ratio(a::AlgebraicQuantity, b::AlgebraicQuantity), a::AlgebraicQuantity, null),

Definition(equal(ratio(A::GeometricQuantity, B::GeometricQuantity), ratio(a::Algebraic-Quantity, b::AlgebraicQuantity)), [equal(times(b, A), times(a, B))], null),

Definition(isout(A::Point, o::Circle), [lt(length(radius(o)), distance(A, center(o)))], null),

Definition(isin(A::Point, o::Circle), [lt(distance(A, center(o)), length(radius(o)))], null),

Definition(isin(O::Point, triangle(A::Point, B::Point, C::Point)), b::Boolean, null),

Definition(collinear(A::Point, B::Point, C::Point), [incident(A, line(B, C))], and(not(is(A, B)), not(is(B, C)))),

Definition(concurrent(l::Line, m::Line, n::Line), [incident(intersection(l, m), n)], and(not(is(l, n)), not(is(m, n)))),

Definition(perpendicular(line(A::Point, B::Point), line(C::Point, D::Point)), b::boolean, null),

Definition(parallel(line(A::Point, B::Point), line(C::Point, D::Point)), b::Boolean, null),

Definition(intersect(l::Line, m::Line), [not(parallel(l, m))], null),

Definition(intersect(l::Line, m::Circle), [lt(distance(center(m), l), length(radius(m)))], null),

Definition(intersect(m::Circle, n::Circle), [lt(distance(center(m), center(n)), plus(length(radius(m)), length(radius(n))))], null),

Definition(sameside(A::Point, B::Point, l::Line), b::Boolean, null),

Definition(sameside(A::Point, B::Point, C::Point), b::Boolean, null),

Definition(differentside(A::Point, B::Point, l::Line), b::Boolean, null),

Definition(differentside(A::Point, B::Point, C::Point), b::Boolean, null),

Definition(endpoint(segment(A::Point, B::Point)), {[A]; [B]}, null),

Definition(vertex(triangle(A::Point, B::Point, C::Point)), {[A]; [B]; [C]}, null),

Definition(orthocenter(triangle(A::Point, B::Point, C::Point)), [concurrentpoint(altitudeline(A, side(B, C)), altitudeline(B, side(A, C)), altitudeline(C, side(A, B)))], null),

Definition(centroid(triangle(A::Point, B::Point, C::Point)), [concurrentpoint(median(A, side(B, C)), median(B, side(A, C)), median(C, side(A, B)))], null),

Definition(barycenter(triangle(A::Point, B::Point, C::Point)), [centroid(triangle(A, B, C))], null),

Definition(geometriccenter(triangle(A::Point, B::Point, C::Point)), [centroid(triangle(A, B, C))], null),

Definition(incenter(t::Triangle), [center(incircle(t))], null),

Definition(circumcenter(t::Triangle), [center(circumcircle(t))], null),

Definition(circumradius(t::Triangle), [radius(circumcircle(t))], null),

Definition(inradius(t::Triangle), [radius(incircle(t))], null),

Definition(midperpendicularline(segment(A::Point, B::Point)), [perpendicularline(midpoint(segment(A, B)), line(A, B))], null),

Definition(distance(A::Point, B::Point), l::Length, null),

Definition(distance(A::Point, l::Line), [distance(A, foot(A, l))], null),

Definition(pointon(o::Circle), [A::Point where incident(A, o)], null),

Definition(pointon(l::Line), [A::Point where incident(A, l)], null),

Definition(midpoint(A::Point, B::Point), [midpoint(segment(A, B))], null),

Definition(midpoint(segment(A::Point, B::Point)), [C::Point where and(equal(distance(C, A), distance(C, B)), incident(C, line(A, B)))], null),

Definition(circle(O::Point, A::Point), [circle(O, distance(O, A))], null),

Definition(circle(A::Point, B::Point, C::Point), [circle(O::Point, A) where and(equal(distance(O, A), distance(O, B)), equal(distance(O, A), distance(O, C)))], null),

Definition(foot(A::Point, l::Line), [intersection(perpendicularline(A, l), l)], not(incident(A, l))),

Definition(intersection(l::Line, m::Line), [A::Point where and(incident(A, l), incident(A, m))], not(parallel(l, m))),

Definition(intersection(l::Line, o::Circle), {[A::Point where and(incident(A, l), incident(A, o))]; [B::Point where and(incident(B, l), incident(B, o))]}, intersect(l, o)),

Definition(intersection(halfline(A::Point, B::Point), o::Circle), [C::Point where and(incident(C, halfline(A, B)), incident(C, o))], in(A, o)),

Definition(intersection(m::Circle, n::Circle), {[A::Point where and(incident(A, m), incident(A, n))]; [B::Point where and(incident(B, m), incident(B, n))]}, intersect(m, n)),

Definition(tangentpoint(l::Line, o::Circle), [A::Point where and(incident(A, l), incident(A, o))], tangent(l, o)),

Definition(tangentpoint(m::Circle, n::Circle), [A::Point where and(incident(A, m), incident(A, n))], tangent(m, n)),

Definition(2tangentline(A::Point, o::Circle), {[l::Line where and(incident(A, l), tangent(l, o))]; [
m::Line where and(incident(A, m), tangent(m, o))]}, out(A, o)),
Definition(1tangentline(A::Point, o::Circle), [perpendicularline(A, line(A, Center(o)))],
incident(A, o)),
Definition(center(circle(O::Point, r::Length)), [O], null),
Definition(radius(circle(O::Point, r::Length)), [segment(O, B::Point) where equal(
distance(O, B), r)], null),
Definition(length(radius(circle(O::Point, r::Length))), [r], null),
Definition(SBintangentcircle(A::Point, o::Circle), [circle(A, B::Point) where
SBintangent(circle(A, B), o)], and(in(A, o), not(is(A, center(o))))),
Definition(BSintangentcircle(A::Point, o::Circle), [circle(A, B::Point) where
BSintangent(circle(A, B), o)], not(is(A, center(o)))),
Definition(extangentcircle(A::Point, o::Circle), [circle(A, B::Point) where
extangent(circle(A, B), o)], out(A, o)),
Definition(tangent(m::Circle, n::Circle), [or(extangent(m, n), intangent(m, n))], null),
Definition(tangentcircle(A::Point, o::Circle), [circle(A, B::Point) where
tangent(circle(A, B), o)], not(is(A, center(o)))),
Definition(extangent(m::Circle, n::Circle), [equal(plus(length(radius(m)),
length(radius(n))), distance(center(m), center(n)))], null),
Definition(intangent(m::Circle, n::Circle), [or(BSintangent(m, n), SBintangent(m, n))], null),
Definition(BSintangent(m::Circle, n::Circle), [equal(minus(length(radius(m)), length(radius(
n))), distance(center(m), center(n)))], null),
Definition(SBintangent(m::Circle, n::Circle), [equal(minus(length(radius(n)), length(radius(
m))), distance(center(m), center(n)))], null),
Definition(side(A::Point, B::Point), [segment(A, B)], null),
Definition(median(A::Point, segment(B::Point, C::Point)), [segment(A, midpoint(segment(B,
C)))], not(collinear(A, B, C))),
Definition(medians(triangle(A::Point, B::Point, C::Point)), [median(A, segment(B, C))];[
median(B, segment(A, C))]; [median(C, segment(A, B))], null),
Definition(medians(triangle(A::Point, B::Point, C::Point)), [median(A, segment(B, C))];[
median(B, segment(A, C))]; [median(C, segment(A, B))], null),

Definition(cevian(A::Point, segment(B::Point, C::Point)), [segment(A, D::Point) where incident(D, segment(B, C))], not(collinear(A, B, C))),

Definition(altitude(A::Point, segment(B::Point, C::Point)), [segment(A, foot(A, line(B, C)))], not(collinear(A, B, C))),

Definition(altitudeline(A::Point, segment(B::Point, C::Point)), [perpendicularline(A, line(B, C))], not(collinear(A, B, C))),

Definition(bisectorline(A::Point, B::Point, C::Point), [line(B, D::Point) where equal(size(angle(C, B, D)), size(angle(D, B, A)))], null),

Definition(bisector(B::Point, A::Point, C::Point), [segment(A, intersection(bisectorline(B, A, C), line(B, C)))], not(collinear(A, B, C))),

Definition(bisectors(triangle(A::Point, B::Point, C::Point)), {[bisector(A, B, C)]; [bisector(B, C, A)]; [bisector(C, A, B)]}, null),

Definition(ninepointcircle(triangle(A::Point, B::Point, C::Point)), [circle(foot(A, line(B, C)), foot(B, line(A, C)), foot(C, line(A, B)))], null),

Definition(Eulerpoint(A::Point, t::Triangle), [midpoint(segment(A, orthocenter(t)))], null),

Definition(Eulerline(a::Triangle), [collinearline(orthocenter(a), centroid(a), circumcenter(a))], null),

Definition(collinearline(A::Point, B::Point, C::Point), [line(choosediff({A; B; C}, 2))], collinear(A, B, C)),

Definition(concurrentpoint(l::Line, m::Line, n::Line), [intersection(choosediff({l; m; n}, 2))], concurrent(l, m, n)),

Definition(concurrentpoint(segment(A::Point, B::Point), segment(C::Point, D::Point), segment(E::Point, F::Point)), [concurrentpoint(line(A, B), line(C, D), line(E, F))], null),

Definition(circumcircle(triangle(A::Point, B::Point, C::Point)), [circle(A, B, C)], null),

Definition(incircle(triangle(A::Point, B::Point, C::Point)), [circle(O::Point, distance(O, line(A, B))) where and(equal(distance(O, line(A, B)), equal(O, line(B, C))), equal(distance(O, line(A, B)), equal(O, line(A, C))), isin(O, triangle(A, B, C)))], null),

Definition(excircle(C::Point, side(A::Point, B::Point)), [circle(O::Point, distance(O, line(A, B))) where and(equal(distance(O, line(A, B)), equal(O, line(B, C))), equal(distance(O, line(A, B)), equal(O, line(A, C))), differentside(A, O, line(B, C)))], null),

Definition(excircle(triangle(A::Point, B::Point, C::Point)), {[excircle(A, side(B, C))]; [excircle(B,

side(A, C))]; [excircle(C, side(A, B))]}, null),
Definition(escenter(C::Point, side(A::Point, B::Point)), [center(excircle(C, side(A, B)))], null),
Definition(escenter(triangle(A::Point, B::Point, C::Point)), {[center(excircle(A, side(B, C)))]; [
center(excircle(B, side(A, C)))]; [center(excircle(C, side(A, B)))]}, null),
Definition(tritangentcircle(triangle(A::Point, B::Point, C::Point)), {[incircle(triangle(A, B, C))
]; [excircle(A, side(B, C))]; [excircle(B, side(A, C))]; [excircle(C, side(A, B))]}, null),
Definition(equilateraltriangle(A::Point, B::Point, C::Point), [triangle(A, B, C) where and(equal
(length(side(A, B)), length(side(A, C))), equal(length(side(A, B)), length(side(B, C))))], null),
Definition(externalequilateraltriangle(segment(A::Point, B::Point), C::Point), [equilateral-
triangle(A, B, D::Point) where differentside(C, D, line(A, B))], not(incidentLine(C, line(A, B)))),
Definition(Napoleontriangle(triangle(A::Point, B::Point, C::Point)), [triangle(barycenter(
externalequilateraltriangle(side(B, C), A)), barycenter(externalequilateraltriangle(side(C, A),
B)), barycenter(externalequilateraltriangle(side(A, B), C)))], null),
Definition(Pappus(A::Point, B::Point, C::Point, D::Point, E::Point, F::Point, P::Point, Q::Point,
R::Point), [configuration(C := pointon(line(A, B)), F := pointon(line(D, E)), P := intersection(
line(A, E), line(D, B)), Q := intersection(line(A, F), line(D, C)), R := intersection(line(B, F), line(E,
C)))], null),
Definition(concyclic(A::Point, B::Point, C::Point, D::Point, E::Point, F::Point), [configuration(
O::Point := center(circle(A, B, C)), incident(D, circle(O, A)), incident(E, circle(O, A)), incident(F,
circle(O, A)))], null),
Definition(concyclic(A::Point, B::Point, C::Point, D::Point), [incident(D, circle(A, B, C))], null),
Definition(completequadrilateral(A::Point, B::Point, C::Point, D::Point, E::Point, F::Point), [con-
figuration(E := intersection(line(A, B), line(C, D)), F := intersection(line(A, C), line(B, D)))], null),
Definition(diagonal(completequadrilateral(A::Point, B::Point, C::Point, D::Point, E::Point, F::
Point)), {[segment(A, D)]; [segment(B, C)]; [segment(E, F)]}, null),
Definition(Gaussline(A::Point, B::Point, C::Point, D::Point), [collinearline(midpoint(diagonal
(completequadrilateral(A, B, C, D, E::Point, F::Point))))], null),
Definition(Simsonline(D::Point, triangle(A::Point, B::Point, C::Point)), [collinearline(foot(D,
side(A, B)), foot(D, side(A, C)), foot(D, side(B, C)))], incident(D, circumcircle(triangle(A, B, C)))),
Definition(mediator(segment(A::Point,B::Point)),[midperpendicularline(segment(A,B))],null),
Definition(orthictriangle(triangle(A::Point, B::Point, C::Point)), [triangle(foot(A, side(B, C)),

foot(B, side(A, C)), foot(C, side(A, B)))], null),

Definition(homothetic(triangle(A::Point, B::Point, C::Point), triangle(A::Point, B::Point, C::Point)), [and(parallel(line(A, B), line(D, E)), parallel(line(A, C), line(D, F)), parallel(line(B, C), line(E, F)))], null),

Definition(homotheticcenter(triangle(A::Point, B::Point, C::Point), triangle(D::Point, E::Point, F::Point)), [concurrentpoint(line(A, D), line(B, E), line(C, F))], null),

Definition(tangentialtriangle(triangle(A::Point, B::Point, C::Point)), [triangle(D::Point, E::Point, F::Point) where and(equal(length(segment(O::Point, A)), length(segment(O, B))), equal(length(segment(O, A)), length(segment(O, C))), perpendicular(line(O, A), line(E, F)), perpendicular(line(O, B), line(D, F)), perpendicular(line(O, C), line(D, E)))], null),

Definition(antiparallel(l::Line,m::Line,n::Line),[equal(size(angle(l,n)),size(angle(n,m)))],null),

Definition(orthorcentricgroup(H::Point, triangle(A::Point, B::Point, C::Point)), [configuration(H := orthocenter(triangle(A, B, C)))], null),

Definition(circumdiameterline(A::Point, side(B::Point, C::Point)), [line(A, circumcenter(triangle(A, B, C)))], null),

Definition(symmetry(A::Point, line(B::Point, C::Point)), [D::Point where and(perpendicular(line(A, D), line(B, C)), equal(distance(A, line(B, C)), distance(D, line(B, C))))], null),

Definition(anticomplementarytriangle(triangle(A::Point, B::Point, C::Point)), [triangle(D::Point, E::Point, F::Point) where and(is(A, midpoint(segment(D, E))), is(B, midpoint(segment(E, F))), is(C, midpoint(segment(D, F))))], null),

Definiiton(anticenter(cyclicquadrilateral(A::Point, B::Point, C::Point, D::Point)), [concurrentpoint(maltitude(quadrilateral(A, B, C, D)))], null),

Definition(maltitude(quadrilateral(A::Point, B::Point, C::Point, D::Point)), {[perpendicularline(midpoint(A, B), line(C, D))]; [perpendicularline(midpoint(B, C), line(A, D))]; [perpendicularline(midpoint(C, D), line(A, B))]; [perpendicularline(midpoint(A, D), line(B, C))]}, null),

Definition(cyclicquadrilateral(A::Point, B::Point, C::Point, D::Point), [quadrilateral (A, B, C, D) where incident(D, circle(A, B, C))], null),

Definition(orthodiagonalquadrilateral(A::Point, B::Point, C::Point, D::Point), [quadrilateral(A, B, C, D) where perpendicular(line(A, C), line(B, D))], null),

Definition(Miquelcircle(A::Point, D::Point, E::Point, side(B::Point, C::Point)), [

circle(A, D, E) where and(incident(D, segment(A, B)), incident(E, segment(A, C)))], null),
Definition(contacttriangle(triangle(A::Point, B::Point, C::Point)), [triangle(foot(
O::Point, line(A, B)), foot(O, line(A, C)), foot(O, line(B, C)))where is(O, incenter(
triangle(A, B, C)))], null),
Definition(parallelogram(A::Point, B::Point, C::Point, D::Point), [quadrilateral(A, B, C, D)
where and(parallel(line(A, B), line(C, D)), parallel(line(A, D), line(B, C)))], null)
)

F. 构造型定义脚本

Script(Definition(point(), A::Point, null),

Definition(line(A::Point, B::Point), a::Line, not(is(A, B))),

Definition(segment(A::Point, B::Point), s::Segment, not(is(A, B))),

Definition(length(segment(A::Point, B::Point)), [distance(A, B)], null),

Definition(halfline(A::Point, B::Point), h::Halfline, not(is(A, B))),

Definition(triangle(A::Point, B::Point, C::Point), t::Triangle, not(collinear(A, B, C))),

Definition(triangle(l::Line, m::Line, n::Line), [triangle(intersection(l, m), intersection(l, n), intersection(m, n))], not(concurrent(l, m, n))),

Definition(quadrilateral(A::Point, B::Point, C::Point, D::Point), a::Quadrilateral, null),

Definition(angle(A::Point, B::Point, C::Point), a::Angle, and(not(is(A, B)), not(is(B, C)), not(is(A, C)))),

Definition(angle(line(A::Point, B::Point), line(C::Point, D::Point)), [angle(A, intersection(line(A, B), line(C, D)), C) where intersect(line(A, B), line(C, D))], null),

Definition(conic(A::Point, B::Point, C::Point, D::Point, E::Point), c::Conic, null),

Definition(size(angle(A::Point, B::Point, C::Point)), s::Degree, null),

Definition(circle(O::Point, r::Length), c::Circle, not(equal(r, 0::Length))),

Definition(arc(A::Point, B::Point, C::Point), a::Arc, null),

Definition(perpendicularline(A::Point, l::Line), m::Line, null),

Definition(parallelline(A::Point, l::Line), m::Line, not(incident(A, l))),

Definition(is(A::Point, B::Point), b::Boolean, null),

Definition(is(line(A::Point, B::Point), l::Line), [and(incident(A, l), incident(B, l))], null),

Definition(is(triangle(A::Point, B::Point, C::Point), triangle(D::Point, E::Point, F::Point)), [and(is(A, D), is(B, E), is(C, F))], null),

Definition(incident(A::Point, o::Circle), [equal(distance(center(o), A), length(radius(o)))], null),

Definition(incident(A::Point, l::Line), b::Boolean, null),

Definition(incident(A::Point, segment(B::Point, C::Point)), [and(incident(A, line(B, C)), differentside(A, B, C))], null),

Definition(incident(A::Point, line(B::Point, C::Point)), b::Boolean, null),

Definition(incident(A::Point, halfline(B::Point, C::Point)), [and(incident(A, line(B, C)), sameside(A, C, B))], null),

Definition(incident(A::Point, perpendicularline(B::Point, l::Line)), [perpendicular(line(A, B), l)], null),

Definition(incident(A::Point, parallelline(B::Point, l::Line)), [parallel(line(A, B), l)], null),

Definition(incident(A::Point, bisector(B::Point, C::Point, D::Point)), [equal(distance(A, line(B, C)), distance(A, line(C, D)))], null),

Definition(ratio(A::GeometricQuantity, B::GeometricQuantity), a::AlgebraicQuantity, null),

Definition(ratio(a::AlgebraicQuantity, b::AlgebraicQuantity), a::AlgebraicQuantity, null),

Definition(equal(ratio(A::GeometricQuantity, B::GeometricQuantity), ratio(a::AlgebraicQuantity, b::AlgebraicQuantity)), [equal(times(b, A), times(a, B))], null),

Definition(isout(A::Point, o::Circle), [lt(length(radius(o)), distance(A, center(o)))], null),

Definition(isin(A::Point, o::Circle), [lt(distance(A, center(o)), length(radius(o)))], null),

Definition(isin(O::Point, triangle(A::Point, B::Point, C::Point)), b::Boolean, null),

Definition(collinear(A::Point, B::Point, C::Point), [incident(A, line(B, C))], and(not(is(A, B)), not(is(B, C)))),

Definition(concurrent(l::Line, m::Line, n::Line), [incident(intersection(l, m), n)], and(not(is(l, n)), not(is(m, n)))),

Definition(perpendicular(line(A::Point, B::Point), line(C::Point, D::Point)), b::boolean, null),

Definition(parallel(line(A::Point, B::Point), line(C::Point, D::Point)), b::Boolean, null),

Definition(intersect(l::Line, m::Line), [not(parallel(l, m))], null),

Definition(intersect(l::Line, m::Circle), [lt(distance(center(m), l), length(radius(m)))], null),

Definition(intersect(m::Circle, n::Circle), [lt(distance(center(m), center(n)), plus(length(radius(m)), length(radius(n))))], null),

Definition(sameside(A::Point, B::Point, l::Line), b::Boolean, null),

Definition(sameside(A::Point, B::Point, C::Point), b::Boolean, null),

Definition(differentside(A::Point, B::Point, l::Line), b::Boolean, null),

Definition(differentside(A::Point, B::Point, C::Point), b::Boolean, null),

Definition(endpoint(segment(A::Point, B::Point)), {[A]; [B]}, null),
Definition(vertex(triangle(A::Point, B::Point, C::Point)), {[A]; [B]; [C]}, null),
Definition(orthocenter(triangle(A::Point, B::Point, C::Point)), [concurrentpoint(
altitudeline(A, side(B, C)), altitudeline(B, side(A, C)), altitudeline(C, side(A, B)))], null),
Definition(centroid(triangle(A::Point, B::Point, C::Point)), [concurrentpoint(median(A,
side(B, C)), median(B, side(A, C)), median(C, side(A, B)))], null),
Definition(barycenter(triangle(A::Point, B::Point, C::Point)), [centroid(triangle(A, B, C))],
null),
Definition(geometriccenter(triangle(A::Point, B::Point, C::Point)), [centroid(triangle
(A, B, C))], null),
Definition(incenter(t::Triangle), [center(incircle(t))], null),
Definition(circumcenter(t::Triangle), [center(circumcircle(t))], null),
Definition(circumradius(t::Triangle), [radius(circumcircle(t))], null),
Definition(inradius(t::Triangle), [radius(incircle(t))], null),
Definition(midperpendicularline(segment(A::Point, B::Point)), [perpendicularline(midpoint
(segment(A, B)), line(A, B))], null),
Definition(distance(A::Point, B::Point), l::Length, null),
Definition(distance(A::Point, l::Line), [distance(A, foot(A, l))], null),
Definition(pointon(o::Circle), A::Point, null),
Definition(pointon(l::Line), A::Point, null),
Definition(midpoint(A::Point, B::Point), [midpoint(segment(A, B))], null),
Definition(midpoint(segment(A::Point, B::Point)), C::Point, null),
Definition(circle(O::Point, A::Point), o::Circle, null),
Definition(circle(A::Point, B::Point, C::Point), o::Circle, null),
Definition(foot(A::Point, l::Line), [intersection(perpendicularline(A, l), l)], not(
incident(A, l))),
Definition(intersection(l::Line, m::Line), A::Point, intersect(l, m)),
Definition(intersection(l::Line, o::Circle), A::Point, intersect(l, o)),
Definition(intersection(halfline(A::Point, B::Point), o::Circle), C::Point, in(A, o)),
Definition(intersection(m::Circle, n::Circle), A::Point, intersect(m, n)),
Definition(tangentpoint(l::Line, o::Circle), A::Point, tangent(l, o)),

Definition(tangentpoint(m::Circle, n::Circle), A::Point, tangent(m, n)),
Definition(2tangentline(A::Point, o::Circle), l::Line, out(A, o)),
Definition(1tangentline(A::Point, o::Circle), [perpendicularline(A, line(A, Center(o)))],
incident(A, o)),
Definition(center(o::Circle), O::Point, null),
Definition(radius(o::Circle), [segment(center(o), pointon(o))], null),
Definition(SBintangentcircle(A::Point, o::Circle), [circle(A, intersection(halfline
(center(o), A), o))], and(in(A, o), not(is(A, center(o))))),
Definition(BSintangentcircle(A::Point, o::Circle), [circle(A, intersection(halfline
(A, center(o)), o))], and(in(A, o), not(is(A, center(o))))),
Definition(extangentcircle(A::Point, o::Circle), [circle(A, intersection(segment(
center(o), A), o))], out(A, o)),
Definition(tangent(m::Circle, n::Circle), [or(extangent(m, n), intangent(m, n))], null),
Definition(tangentcircle(A::Point, o::Circle), [circle(A, intersection(line(center(o),
A), o))], not(is(A, center(o)))),
Definition(extangent(m::Circle, n::Circle), [equal(plus(length(radius(m)), length(radius
(n))), distance(center(m), center(n)))], null),
Definition(intangent(m::Circle, n::Circle), [or(BSintangent(m, n), SBintangent(m, n))], null),
Definition(BSintangent(m::Circle, n::Circle), [equal(minus(length(radius(m)), length(
radius(n))), distance(center(m), center(n)))], null),
Definition(SBintangent(m::Circle, n::Circle), [equal(minus(length(radius(n)), length(
radius(m))), distance(center(m), center(n)))], null),
Definition(side(A::Point, B::Point), [segment(A, B)], null),
Definition(median(A::Point, segment(B::Point, C::Point)), [segment(A, midpoint(segment
(B, C)))], not(collinear(A, B, C))),
Definition(cevian(A::Point, segment(B::Point, C::Point)), [segment(A, D::Point)where
D := pointon(segment(B, C))], not(collinear(A, B, C))),
Definition(altitude(A::Point, segment(B::Point, C::Point)), [segment(A, foot(A, line
(B, C)))], not(collinear(A, B, C))),
Definition(altitudeline(A::Point, segment(B::Point, C::Point)), [perpendicularline
(A, line(B, C))], not(collinear(A, B, C))),

Definition(bisectorline(A::Point, B::Point, C::Point), l::Line, null),
Definition(bisector(B::Point, A::Point, C::Point), [segment(A, intersection(bisectorline
(B, A, C), line(B, C)))], not(collinear(A, B, C))),
Definition(bisectors(triangle(A::Point, B::Point, C::Point)), {[bisector(A, B, C)]; [bisector(B, C,
A)]; [bisector(C, A, B)]}, null),
Definition(ninepointcircle(triangle(A::Point, B::Point, C::Point)), [circle(midpoint
(A, B), midpoint(A, C), midpoint(B, C))], null),
Definition(Eulerpoint(A::Point, t::Triangle), [midpoint(segment(A, orthocenter(t)))], null),
Definition(Eulerline(a::Triangle), [collinearline(orthocenter(a), centroid(a), circumcenter
(a))], null),
Definition(collinearline(A::Point, B::Point, C::Point), [line(choosediff({A; B; C}, 2))],
collinear(A, B, C)),
Definition(concurrentpoint(l::Line, m::Line, n::Line), [intersection(choosediff({l; m; n},
2))], concurrent(l, m, n)),
Definition(concurrentpoint(segment(A::Point, B::Point), segment(C::Point, D::Point),
segment(E::Point, F::Point)), [concurrentpoint(line(A, B), line(C, D), line(E, F))], null),
Definition(circumcircle(triangle(A::Point, B::Point, C::Point)), [circle(A, B, C)], null),
Definition(incircle(triangle(A::Point, B::Point, C::Point)), [circle(O::Point, foot(O,
line(A, B))) where O := intersection(bisectorline(A, B, C), bisectorline(B, C, A))], null),
Definition(excircle(C::Point, side(A::Point, B::Point)), [circle(O::Point, foot(O, line(
A, B))) where O := intersection(bisectorline(C, B, A), bisectorline(B, A, C))], null),
Definition(escenter(C::Point, side(A::Point, B::Point)), [intersection(bisectorline
(C, B, A), bisectorline(B, A, C))], null),
Definition(escenter(triangle(A::Point, B::Point, C::Point)), {[escenter(A, side(B, C))
]; [escenter(B, side(A, C))]; [escenter(C, side(A, B))]}, null),
Definition(excircle(triangle(A::Point, B::Point, C::Point)), {[excircle(A, side(B, C))
]; [excircle(B, side(A, C))]; [excircle(C, side(A, B))]}, null),
Definition(tritangentcircle(triangle(A::Point, B::Point, C::Point)), {[incircle(
triangle(A, B, C))]; [excircle(A, side(B, C))]; [excircle(B, side(A, C))]; [excircle(C,
side(A, B))]}, null),
Definition(equilateraltriangle(A::Point, B::Point, C::Point), [triangle(A, B, rotate

```
(A, B, pi/3))], null),
Definition(externalequilateraltriangle(segment(A::Point, B::Point), C::Point), [triangle
(A, B, rotate(A, B, pi/3))], not(incidentLine(C, line(A, B)))),
Definition(Napoleontriangle(triangle(A::Point, B::Point, C::Point)), [triangle(barycenter
(externalequilateraltriangle(side(B, C), A)), barycenter(externalequilateraltriangle(side
(C, A), B)), barycenter(externalequilateraltriangle(side(A, B), C)))], null),
Definition(Pappus(A::Point, B::Point, C::Point, D::Point, E::Point, F::Point, P::Point,
Q::Point, R::Point), [configuration(C := pointon(line(A, B)), F := pointon(line(D, E)),
P := intersection(line(A, E), line(D, B)), Q := intersection(line(A, F), line(D, C)),
R := intersection(line(B, F), line(E, C)))], null),
Definition(concyclic(A::Point, B::Point, C::Point, D::Point, E::Point, F::Point), [
configuration(O::Point := center(circle(A, B, C)), D := pointon(circle(O, A)),
E := pointon(circle(O, A)), F := pointon(circle(O, A)))], null),
Definition(concyclic(A::Point, B::Point, C::Point, D::Point), [incident(D, circle(A, B, C))],
null),
Definition(completequadrilateral(A::Point, B::Point, C::Point, D::Point, E::Point, F::Point), [
configuration(E := intersection(line(A, B), line(C, D)), F := intersection(line(A, C), line(B, D)))],
null),
Definition(diagonal(completequadrilateral(A::Point, B::Point, C::Point, D::Point, E::Point,
F::Point)), {[segment(A, D)]; [segment(B, C)]; [segment(E, F)]}, null),
Definition(Gaussline(A::Point, B::Point, C::Point, D::Point), [collinearline(midpoint(
diagonal(completequadrilateral(A, B, C, D, E::Point, F::Point))))], null),
Definition(Simsonline(D::Point, triangle(A::Point, B::Point, C::Point)), [collinearline
(foot(D, side(A, B)), foot(D, side(A, C)), foot(D, side(B, C)))], null),
Definition(mediator(segment(A::Point, B::Point)), [midperpendicularline(segment(A, B))],
null),
Definition(orthictriangle(triangle(A::Point, B::Point, C::Point)), [triangle(foot(A, line
(B, C)), foot(B, line(A, C)), foot(C, line(A, B)))], null),
Definition(homothetic(triangle(A::Point, B::Point, C::Point), triangle(A::Point, B::Point,
C::Point)), [and(parallel(line(A, B), line(D, E)), parallel(line(A, C), line(D, F)), parallel
(line(B, C), line(E, F)))], null),
```

Definition(homotheticcenter(triangle(A::Point, B::Point, C::Point), triangle(D::Point, E::Point, F::Point)), [concurrentpoint(line(A, D), line(B, E), line(C, F))], null),
Definition(tangentialtriangle(triangle(A::Point, B::Point, C::Point)), [triangle(perpendicularline(A, line(O::Point, A)), perpendicularline(B, line(O, B)), perpendicularline (C, line(O, C))) where O := circumcenter(triangle(A, B, C))], null),
Definition(antiparallel(l::Line, m::Line, n::Line), [equal(size(angle(l, n)), size(angle (n, m)))], null),
Definition(orthorcentricgroup(H::Point, triangle(A::Point, B::Point, C::Point)), [configuration(H := orthocenter(triangle(A, B, C)))], null),
Definition(circumdiameterline(A::Point, side(B::Point, C::Point)), [line(A, circumcenter(triangle(A, B, C)))], null),
Definition(symmetry(A::Point, line(B::Point, C::Point)), [symmetry(A, foot(A, line(B, C)))], null),
Definition(symmetry(A::Point, B::Point), C::Point, null),
Definition(anticomplementarytriangle(triangle(A::Point, B::Point, C::Point)), [triangle (D::Point, E::Point, F::Point) where and(is(A, midpoint(segment(D, E))), is(B, midpoint(segment(E, F))), is(C, midpoint(segment(D, F))))], null),
Definition(anticenter(cyclicquadrilateral(A::Point, B::Point, C::Point, D::Point)), [concurrentpoint(maltitude(quadrilateral(A, B, C, D)))], null),
Definition(maltitude(quadrilateral(A::Point, B::Point, C::Point, D::Point)), {[perpendicularline(midpoint(A, B), line(C, D))]; [perpendicularline(midpoint(B, C), line(A, D))]; [perpendicularline(midpoint(C, D), line(A, B))]; [perpendicularline(midpoint(A, D), line(B, C))]}, null),
Definition(cyclicquadrilateral(A::Point, B::Point, C::Point, D::Point), [quadrilateral (A, B, C, pointon(circle(A, B, C)))], null),
Definition(orthodiagonalquadrilateral(A::Point, B::Point, C::Point, D::Point), [quadrilateral(A, B, C, pointon(perpendicularline(B, line(A, C))))], null),
Definition(Miquelcircle(A::Point, D::Point, E::Point, side(B::Point, C::Point)), [circle(A, pointon(line(A, B)), pointon(line(A, C)))], null),
Definition(contacttriangle(triangle(A::Point, B::Point, C::Point)), [triangle(foot (O::Point, line(A, B)), foot(O, line(A, C)), foot(O, line(B, C)))where O := incenter(triangle

(A, B, C))], null),

Definition(rotate(A::Point, B::Point, d::Degree), C::Point, not(is(A, B))),

Definition(parallelogram(A::Point, B::Point, C::Point, D::Point), [quadrilateral(A, B, C, D)

where D := intersection(parallelline(A, line(B, C)), parallelline(C, line(A, B)))], null)

)

攻读博士学位期间发表和完成的论文

[1] Chen X. and Wang D. Management of Geometric Knowledge in Textbooks[J]. Data & Knowledge Engineering (DKE), 2011, in press

[2] Chen X. Electronic Geometry Textbook: A Geometric Textbook Knowledge Management System[A]. Autexier S., Calmet J. and Delahaye D. (eds.), Proceedings of the 9th International Conference on Mathematical Knowledge Management (MKM 2010)[C]. LNAI **6167**, Springer-Verlag, Berlin Heidelberg, 2010:278-292

[3] Chen X., Huang Y. and Wang D. On the Design and Implementation of a Geometric knowledge Base[A]. Sturm T. and Zengler C. (eds.), Proceedings of the 7th International Workshop on Automated Deduction in Geometry (ADG 2008)[C]. LNAI **6301**, Springer-Verlag, Berlin Heidelberg, 2011:22-41

[4] Chen X. and Wang D. Towards an Electronic Geometry Textbook[A]. Botana F. and Recio T. (eds.), Post-proceedings of the 6th International Workshop on Automated Deduction in Geometry (ADG 2006)[C]. LNAI **4869**, Springer-Verlag, Berlin Heidelberg, 2007:1-23

[5] Chen X. Formal Representation and Automated Transformation of Geometric Statements[A]. Richter-Gebert J. and Schreck P. (eds.), Proceedings of the 8th International Workshop on Automated Deduction in Geometry (ADG 2010)[C]. Technical University Munich, Germany, 2010:1-19

[6] Chen X., Liang T., Wang D. and Zhao T. Towards a Dynamic Environment for Geometry Research & Education (Extended Abstract)[A]. Ida T., Jiang Q. and Wang D. (eds.), Proceedings of the 5th Asian Workshop on Foundations of Software (AWFS 2007)[C]. Xiamen, China, 2007:153-156

[7] Chen X. and Wang D. Towards an Electronic Geometry Textbook (Extended Abstract)[A]. Botana F. and Roanes-Lozano E. (eds.), the 6th International Workshop on Automated Deduction in Geometry (ADG 2006)[C]. Universidade de Vigo, Pontevedra, Spain, 2006:15-25

[8] 王东明, 黄荧, 陈肖宇. 几何知识库的设计与实现[J]. 计算机应用, 2009, V29(2): 398-402

[9] Chen X. Formalization and Automated Transformation of Geometric Statements [J]. Journal of Systems Science and Complexity (JSSC), 2010, under review

致　谢

首先我要特别感谢导师王东明教授在我博士学习期间对我的悉心培养、谆谆教诲、无私关怀与帮助, 使我一点一点地进步, 一步一步地成长. 王老师渊博的学识和对研究课题高屋建瓴的把握为我在学术上指明了方向. 王老师严谨的科研作风和对工作的满腔热情深深地感染着我, 是我学习的楷模. 更重要的是从王老师身上学到的为人处事的道理和对待生活以及人生的态度将使我受益终身. 我还要特别感谢王老师为我创造多次出国机会, 使我能够亲身感受国外同行的工作状态和做研究的方式, 与他们交流, 向他们学习. 这些经历是我人生中不可多得的宝贵财富.

其次, 我要特别感谢荷兰埃因霍温科技大学的 Arjeh M. Cohen 教授和 Hans Cuypers 博士邀请并资助我访问 DAM 小组. 在一年的访问期间, 良好的学术气氛和舒适的工作环境使我可以安心地做研究, 享受编程的乐趣. 感谢 DAM 小组 Jan Willem Knopper 等成员, 他们的帮助使我在友好的氛围中学习到了很多学术方面的知识.

我要感谢奥地利符号计算研究所的 Bruno Buchberger 教授和 Franz Winkler 教授为我访问提供资助. 与 Bruno Buchberger 教授的几次交谈使我有幸感受到了学术大家的风范, 他对工作的态度和对事业的追求使我受益良多; 感谢美国的 Hoon Hong 教授和法国的 Renaud Rioboo 教授, 他们对我的研究提出了很多宝贵的建议; 感谢李未院士以及数理逻辑讨论班的成员们, 从他们那里我学习到了很多数理逻辑方面的知识.

感谢众位师兄弟姐妹, 和他们在一起学习生活使我收获很多; 感谢在我的生活中给我关心和帮助的同学和朋友, 他们的友善和热情使我获益.

感谢北航国际交流合作处资助我出国参加学术会议; 感谢数学、信息与行为教育部重点实验室等对我学业的支持.

最后我要感谢我的父母, 正是他们无微不至的关爱和鼓励, 使我能够克服生活和学术上的种种困难, 最终完成博士学业.

作者简介

陈肖宇, 男, 汉族, 1982 年 10 月 26 日出生, 辽宁沈阳人. 主要学习与工作经历如下.

2000 年 9 月至 2004 年 7 月就读于北京航空航天大学理学院数学系信息与计算科学专业, 本科.

2004 年 9 月至今就读于北京航空航天大学数学与系统科学学院, 攻读基础数学专业博士学位, 北京航空航天大学数学、信息与行为教育部重点实验室成员, 研究方向为数学知识管理.

2006 年 2 月至 4 月访问奥地利符号计算研究所 (RISC), 访问学生.

2007 年 6 月访问荷兰埃因霍温科技大学 (TU/e) 数学与计算机科学系离散代数和几何 (DAM) 小组, 访问学者.

2008 年 9 月至 2009 年 9 月访问荷兰埃因霍温科技大学 (TU/e) 数学与计算机科学系离散代数和几何 (DAM) 小组, 客座研究员.

担任第三届国际知识管理与信息共享大会 (KMIS 2011, 巴黎) 程序委员会委员, 第八届国际人工智能与符号计算大会 (AISC 2006, 北京) 和第四届计算机与信息科学中的数学方法国际会议 (MACIS 2011, 北京) 地方组委会委员.