电子科技大学

UNIVERSITY OF ELECTRONIC SCIENCE AND TECHNOLOGY OF CHINA

专业学位硕士学位论文

MASTER THESIS FOR PROFESSIONAL DEGREE

论文题目 基于图同构的数学推理引擎的设计与实现

专业学位类别	理学硕士
学　　号	201852081006
作 者 姓 名	李菲
指 导 教 师	符红光　教授

分类号 _____ 密级 _____

UDC[注1] _____

学 位 论 文

基于图同构的数学推理引擎的设计与实现

（题名和副题名）

李菲

（作者姓名）

指导教师 符红光 教授

电子科技大学 成都

（姓名、职称、单位名称）

申请学位级别 硕士 专业学位类别 理学硕士

提交论文日期 _____ 论文答辩日期 _____

学位授予单位和日期 电子科技大学 年 月

答辩委员会主席 _____

评阅人 _____

注 1：注明《国际十进分类法 UDC》的类号。

A Research of Multi-function Switched-beam Antenna Array

A Master Thesis Submitted to

University of Electronic Science and Technology of China

Discipline:	**Master of Pharmacy**
Author:	**Li Fei**
Supervisor:	**Prof. Li Si**
School:	**School of uestc**

摘　要

近些年，人工智能已经由传统的感知智能逐渐向认知智能阶段过渡，认知智能与自动推理成为很多专家学者以及科研机构研究的重点。目前是处于认知智能发展的初期，面临着诸多挑战的同时也伴随着机遇。如何将深度学习应用于逻辑推理，从而让机器具备思考和推理能力将是人工智能重大突破口。本文的研究内容是基于图同构的初等数学推理引擎的设计和构建，推理引擎是整个自动推理系统最核心的部分。推理引擎的系统设计理念是基于产生式系统，同样涉及到知识表示和实例化规则库构建两部分。具体研究内容如下：

（1）初等数学的知识表示

知识表示是类人解答系统求解问题的第一步，只有先将人类的语义转换成计算机的知识结构，才能进行后续的推理和问题求解工作。知识表示的方法多种，由于本文研究课题对象是初等数学问题，分析和总结初等数学的知识具有概念清晰严谨、定理明确以及公式化程度高等特点，最终选用知识图谱来表示数学中概念实体和它们之间的关系在实际编码中使用 Java 中的类结构来构建初等数学的实体类和关系类，为了知识图谱构建的可扩展性和开闭性原则，选用抽象类的继承思想去扩展，即所有的实体类都继承于抽象实体类，所有的关系类都继承于抽象关系类。

（2）构建实例化定理库

基于产生式系统的推理引擎，规则库是引擎重要的外部驱动和产生知识的依据。本文构建的规则库也是知识图谱的方式表示和存储。实例化定理库的构建分为三个步骤：定理来源、定理标准化以及定理实例化。首先定理来源于教材和教辅中的公式定理，以及标准答案的解答过程、解题技巧。对于搜索的定理是需要转换成推理引擎统一的格式，这个过程叫定理标准化，方便推理引擎简单统一。最后将标准化定理生成知识图谱完成实例化。

（3）图同构的推理引擎的设计和构建本文的推理引擎设计思想上可以从三个不同的角度去理解。首先是引擎中推理系统设计模式，是基于”先逆后正“逻辑的产生式系统，并设计成逻辑推理与计算推理交互推理的混合推理；其次，从引擎逻辑结构考虑，引擎采用分层结构，每个逻辑层之间保持相对独立性；最后，考虑引擎核心算法的设计，图匹配的算法思想在于是一种混合匹配算法，知识图谱上通过类型匹配构建实体和关系的映射，然后再对字符串做模式匹配，生成符号轮换集。

本文完成了基于图同构的数学推理引擎的整个算法设计和所有模块的构建工作，完成非应用题的随机批量测试和高考试卷测试，综合解题率 71.2%, 平均求解时间不超过 5 分钟。

关键词： 认知智能，图匹配推理引擎，符号计算，类人解答系统，知识表示，实例化定理

ABSTRACT

With the widespread engineering applications ranging from broadband signals and non-linear systems, time-domain integral equations (TDIE) methods for analyzing transient electromagnetic scattering problems are becoming widely used nowadays. TDIE-based marching-on-in-time (MOT) scheme and its fast algorithm are researched in this dissertation, including the numerical techniques of MOT scheme, late-time stability of MOT scheme, and two-level PWTD-enhanced MOT scheme. The contents are divided into four parts shown as follows.

Keywords: time-domain electromagnetic scattering, time-domain integral equation (TDIE), marching-on in-time (MOT) scheme, late-time instability, plane wave time-domain (PWTD) algorithm

目 录

第一章 绪 论

1.1 研究工作的背景与意义

自 1956 年人工智能概念提出以来，全世界的专家学者都一直致力于人工智能不同领域的研究[1]。经过长达半个多世纪的发展，它的发展历程可谓是几经波折，经过短暂辉煌，也同样出现过很多质疑和冷落。从开始的被质疑到后期被理论证明再到最后的广泛应用，人工智能经历了三个不同阶段，最起初属于机器智能时代，早期最为新兴学科吸引越来越多的科研人员参与研究，很快相继出现了一批显著的成果，如机器化定理证明[2]、跳棋程序、求解程序等，但由于机器翻译等应用的失败，人工智能进入了发展的停滞期。后面神经网络飞速发展，人工智能进入到感知智能时代，各种不同的神经网络模型加上搭载一系列的传感器应用于不同领域，人脸识别、多任务分类、图像处理等领域取得成果有目共睹。随后，人工智能进入到第三阶段，认知智能。所谓认知智能，即机器具备像人类一样具有独立思考、理解、以及逻辑推理等认知能力，机器学会逻辑推理是人工智能发展的终极目标。认知智能的潜在应用领域非常广泛，无人驾驶、智慧医疗、金融风控、机器老师等领域都急需认知智能取得突破。

近些年来，人工智能捷报频传。在金融风控和交易领域，利用其强大的计算能力和存储能力，在银行股市等行业它将取代人们成为新的交易员。相比于人类交易员，计算机不仅能出色完成数据管理和计算，还能依据大数据挖掘技术进行风险预测和价值评估。根据美国花旗银行的最新研究报告，从 2012 年截至 2014 年底人工智能投资管理顾问的资产增值量接近 140 亿美元[3]。在后面的几年时间内这个投资管理的资产额度以指数级别增长，预计总额度将达到 5 万亿美元。在竞技体育方面，人工智能同样大放异彩。Edge Up Sports 与 IBM 团队合作，前者提供了球员的上场时间、进球数、每场触球平均时长，以及比赛场地、时间、天气等外在客观因素等所有能影响比赛的数据，IBM 则利用这些大数据使用相关模型进行大数据分析和预测比赛趋势。同样通过大数据挖掘可以为教练员提供更直观的每个球员每场的表现，从而为每个球员制定不同的训练计划表。无人驾驶领域更是当前热门的研究领域，各大公司和科研都想在无人驾驶方面能取得突破性成果。无论是谷歌的无人驾驶汽车已经完成了在美国各种不同城市不同路况下测试，虽然也会发生一些判断失准导致的一些事故问题，还是国内的百度人工智能研究院的智能汽车，小马智行等公司相继投入无人驾驶的研发中。人工智能还有很多成熟的应用场景，比如个人助手，苹果的 sari，微软的 cortana 都是人工智能在语音

的应用。

人工智能初期阶段的应用之一就是机械化定理证明，随着人工智能不断的发展并在各个领域取得显著的成绩时，却发现机械化定理证明多年没有实质性进展，人工智能在认知推理方面表现的不如人意。事实上，几何机械化问题最早可以追溯到 17 世纪著名大数学家、大思想家莱布尼茨就已经提出机械化证明定理的想法，但莱布尼茨并没有将这种设想形成数学形式。直到 19 世纪大数学家希尔伯特及其同时代的其他数学家完成数理逻辑的创立和发展，从而完成了几何定理数学化的形式。随着 20 世纪中期计算机的出现和发展，才让定理机械化证明有了实际可以实现的可能。从最开始希尔伯特的著作《几何基础》中提出“对于可机械化证明的定理 H 和 P，需要满足的特征是部分的代数关系是对于局部变量必须线性”，再到 20 世纪 30 年代，美国数学家 J.F.Ritt 提出了代数几何的构造性理论[4]，这一理论对后面很多数学工作者具有启发式意义，其中包括我国著名数学家吴文俊和他的“吴方法”也受他的理论影响。“吴方法”是 20 世纪 70 年代末吴文俊先生在中国传统机械化思想的影响下，并研究了 Ritt 等人的理论工作，针对几何定理机器证明问题给出了新的方法[5]，该方法具有高效性和实用性，能够切实应用计算机上进行程序式验证。通过大量的机器证明，吴方法在国际上引起了很大的轰动，从而掀起了机器证明新的高潮。

近些年，随着教育资源的分配不公化、不均匀，教育成本过高等一些列问题的暴露，如何将人工智能应用于教育培训领域成为新的热门话题。人工智能赋能于教育是人们开始追寻和探索的方向。“人工智能 + 教育“具体有五个方面的应用，第一阶段是自动搜题，可以帮组学生从图库中搜索原题的解答；第二阶段是相似题推荐，主要是依赖于深度学习模型和相似度比对算法，在题目中寻找相似题，可以让学生加强巩固同类型题目；第三阶段知识点推荐，主要依赖于推荐算法，对学生和老师进行精准的知识点推荐，老师减轻备课的压力，学生可以有重点的学习；第四阶段：自动判卷，辅助老师去批改学生作业并生成作业批改报告；第五阶段：机器老师。机器老师的核心是自动求解题目，而自动求解的核心便是自动推理，是认知智能的范畴，然后融合前四个阶段的应用，可以是一个于学生可交互的应用于解题、讲题、推荐于一身的机器人。前三个阶段目前均有不同程度的应用，第四、五阶段目前还是处于研究实验状态。本文的研究内容重点就是第五阶段自动解题系统的构建和设计，希望此推理系统可以在人工智能应用于教育领域发挥一定的作用。

1.2 研究现状

1.2.1 知识图谱研究现状

在新一代人工智能发展阶段，即认知智能阶段。由于专家系统的出现，让人们重新认识到知识的重要性。人类的语义包含丰富的信息，这些语义信息可以转换成知识进行传递和交流。而人工智能的大数据挖掘还是处于低纬度的特征空间，这就导致机器学习与人类语义之间存在鸿沟。如何让机器去理解人类语义，最重要的是解决知识表示问题，而计算机知识表示需要载体，知识图谱的出现很好起到语义信息到知识表示的桥梁。知识图谱可以承载非常庞大的语义系统，存储是以图的形式来表示知识，图的基本构成是点 (Point) 和边 (Edge)，而点表示的是一种概念”实体“，例如一些名词概念”人“、”动物“、“天空”、“海洋”等，边表示的是这些概念实体之间的关联，即“关系”，例如概念实体“老虎”与概念实体“动物”之间的关系为“属于关系”，这样通过关系来描述所有实体之间的关联，知识图谱是一张描述不同概念实体之间关联巨大网络图。知识图谱的优势在于将数据的粒度从文本形式降到 data 形式，有利于知识的形式化表示和快速的搜索查询。因此可以基于知识图谱做知识分类、知识聚合、知识推理等一系列任务。

知识图谱从最早谷歌提出概念到今天近 10 年的时间，它的发展和理论研究都是一路高歌猛进，同时知识图谱应用也非常广泛，已经渗透到各行各业以及不同的学科领域。最早应用于语义搜索领域，语义搜索是被称为互联网之父的 Tim Berners-Lee 在 2001 年《科学美国人》（Scientific American）上发表的一篇文章中首次提出的概念[6,7]。在这篇文章他解释了语义搜索的本质含义。“语义搜索的本质是不在是搜索引擎输入框中输入的表层含义，它是可以进行深层搜索的一种深层知识关联度的查询，其中包括联想搜索和同义词语义替换”。传统的语义搜索存在搜索响应慢、无法进行同义词搜索、无法深层搜索等问题，这些问题的根本原因在于知识表示不够完善。因此谷歌才提出知识图谱的概念，试图使用知识图谱来表征语义知识，将搜索引擎的算法迁移到对图上语义的搜索，图表示语义更加准确、信息化更加丰富。同时图的搜索效率相比于之前的文本搜索会提高不少，可以很好的优化搜索效率，提高响应速度。知识图谱的搜索过程表现为：通过理解用户的输入描述，抽取描述中的实体和关系，在去匹配知识图谱中的实体和关系，寻找一定的关联度信息，将查询的结果再重新返回给用户。在问答系统 (QA) 领域，所谓智能问答是指通过问答式对话，让机器具备像人类一样常识知识的理解并完成交互。不同的智能问答系统，它具有不同的功能特点，依赖的知识图谱库也有所不同。例如聊天机器人的知识图谱库就是来源各种常识、百科等构建起来的，机器老师的知识图谱库就来源于数学概念、定理、公理等。这是行业知识

图谱，可以为行业内人提供专业快速的知识查询响应。问答系统时语义搜索的一种延伸，在具备知识图谱的搜索功能推理，还需要考虑上下文的语境，和问答的逻辑性。在个性化推荐领域，根据用户的浏览和消费记录产生的数据构建商品和消费者以及商家之间的关联，生成知识图谱库，再利用基于知识图谱的推荐算法为用户推荐感兴趣的产品或者内容。

1.2.2 推理引擎研究现状

在任何的推理系统中，推理引擎都是系统完成推理功能和驱动的核心组成部分。推理引擎又称推理机，通常情况下推理机由调度器，执行器和一致协调器三部分组成[8,9]。调度器是负责控制引擎的整体流程走向，不同逻辑层和各个模块之间的连通性，相当于推理引擎的大脑，可以起到决策和判断作用。执行器可以视为一组动作集，动作集包括原始事实库的读取和插入，分支选取策略的执行，循环遍历和搜索的执行。一致协调器是为了保证生成的知识保持前后一致，不会出现丢失、改写、被覆盖等情形。规则推理引擎是基于产生式系统思想构建的，由于其逻辑结构清晰，规则独立，编程友好等优点逐渐成为主流设计框架。其核心思想在于将匹配规则单独抽离出来形成规则脚本或者存入到数据库中，使得规则的修改、添加、删除都不需要改动业务逻辑本身，引擎会对已知事实与规则库进行匹配搜索，产生的新知识继续添加到已知事实。

从上世纪 70 年代初开始，斯坦福大学使用 LISP 语言研发了世界上第一个规则引擎-MYCIN 系统[10]，这个系统是最早的规则引擎的雏形，它首次提出将规则知识从系统中抽取出来，通过热插拔的方式去加载规则，提高了系统的灵活性和开放性。该系统在实际应用中也取得了不错的效果。20 世纪 90 年代，随着面向对象技术的兴起，为了解决大型应用软件系统的复杂度问题，开始引入对象封装思想，分类思想，以及跨平台通信机制，同时也为规则的独立性和外部程序封装，以及跨平台交互提供了实现的技术支撑。面向对象的编程思想很好解决规则作为一个个的独立对象存在，可以将规则的所有信息封装成对象，然后独立于业务流程之外，方便规则的扩展。21 世纪后，规则管理技术日趋成熟，2000 年 11 月，JavaCommunity Process 组织开始着手起草 Java 规则引擎的 API 标准，即 JSR94 规范[11,12]。在 JSR94 完成正式定稿的同时，市场上相继推出支持 JSR94 规范的规则引擎[13]，有商用和社区开源引擎，较为成熟的商用引擎代表 ILOG Jrules 和开源引擎代表 Drools，它们的共同点在于引擎稳定高效，规则匹配速度快，从而迅速得到广泛应用。

时至今日，规则引擎也面临一些挑战和问题。第一方面，规则标准化配置问

题。大部分的规则库都是后端相应开发人员构建的形如 xml 格式的配置文件，这些配置文件需要较高的专业水平才能书写，行业外的人很难参与规则库构建，如何让规则库构建更见简易化和方便，让不具备专业知识的人也能参与构建是需要解决的问题。第二方面，规则的维护。需要给每个规则设置唯一的标签，通过唯一的标签快速定位到需要修改、删除、完善的规则。同时需要保持规则之间的独立性，防止删除一个规则导致别的规则无法使用。第三方面，引擎的推理效率。这里效率问题主要是时间开销，当规则库中规则越来越多的时候，引擎对规则库采用盲目式搜索的时间成本会提升，因此对规则库进行分类和适当减枝是非常有必要的，而减枝策略选取成为研究的重点。

1.3 研究内容

本论文主要研究基于初等数学的类人解题系统中推理引擎的设计和实现，通过对主流的一些开源引擎调研，对比分析不同推理引擎的优缺点和它们的设计思想，再结合本文中独有的知识图谱表示法，最终设计和构建了一个同时融合逻辑推理和计算推理的图推理引擎。

对于完整的类人解答系统来说，必须包含四个部分：自然语言理解，知识表示，自动推理和求解，类人解答过程输出。自然语言理解是处理人类的文本描述中所包含的语义，将这种语义理解成计算机可以处理的形式，因此这个部分是这个求解系统的最基础部分，后面的知识表示和自动推理都依赖于自然语言理解的结果。对于知识表示是系统关键问题，它是衔接自然语言理解和自动推理的桥梁，同时知识表示的合理性和简洁性对后续的推理至关重要。表示的过于简单会无法正确表示语义，会出现信息丢失的情况；表示的过于复杂，一方面会导致知识冗余现象，同时会急剧增加推理的复杂度。因此，最终选用知识图谱来表示知识，知识图谱表示知识相对于传统的谓词逻辑的最大优点在于图结构可以表示很复杂的语义结构，并且寻找任何两个实体之间的关联可以转化成图路径算法，同时知识图谱便于展示知识特征，可以很清晰的看出所有的实体和关系。

自动推理和求解部分最核心是图推理引擎的设计。图推理引擎是针对于知识图谱的独立自主开发的推理引擎。从推理方式来说，它融合有逻辑推理和计算推理；本身的逻辑架构来说，引擎包含四层设计结构：图结构层、图匹配层、参数置换层、知识更新层；从设计思想来说，引擎中采用是产生式系统的模式，即对于一组已知事实集合，将已知事实与规则库中的知识进行不断匹配，从而产生新知识，在将产生的新知识重新插入到原始的事实集合中，进而对已知事实集合进行新的迭代，直至到达求解目标终止迭代。

综上所述，本文研究的内容具体可以概括成以下五个部分。

(1) 图推理引擎中所使用的逻辑推理方式

常见的逻辑推理方式分为三种：正向推理、逆向推理、双向推理[14,15]。其中，正向推理是，从已知事实出发，通过匹配规则库产生新知识，然后不断迭代的过程，是一种盲目式搜索的推理方式；而逆向推理恰好相反，它是从目标结论出发，是不断规约的过程，从结论一直往前搜索已有的规则，直到满足所有条件停止规约，是搜索解题路径的一个过程；双向推理则是融合正向推理和逆向推理的混合推理模式，主要包含三种形式：“先正后逆”、“先逆后正”、“正逆同时”。本文的图推理引擎采用是“先逆后正”的方式，即先对已知事实进行逆向搜索去寻找解题路径，然后再使用正向推理对解题路径中所涉及到的规则进行遍历，从盲目是匹配转变成精确匹配规则库中特定规则，很大程度提升了匹配效率。

(2) 逻辑推理与计算推理的交互

因为本文研究的是初等数学问题，必不可少其中涉及很多的数学公式定理的演算，这部分需要符号计算平台提供计算服务。当对定理公式计算时，同样会产生新知识，如何将计算新知识与逻辑推理产生的新知识进行交互，是图推理引擎的一个重要研究内容。最终引擎设计的模式是二者产生的知识相处融合，相互迭代，让推理的方式更加的灵活，也增强了推理引擎的求解能力。

(3) 图推理引擎的逻辑结构和核心算法设计

图推理引擎最大的创新点和亮点在于该引擎是针对于图上的知识表示而非谓词逻辑并且将传统的逻辑推理应用于图上。因此传统基于谓词表示法的推理引擎中的匹配模式就会失效。本文构建的图推理引擎最核心的算法思想是图匹配，即如何判断子图匹配的算法。逻辑架构也是基于图匹配的核心思想搭建的四层结构，每一层之间既能保持相对独立，同时又存在一定的依赖和交互。虽然采用了分层的结构，但在运行的过程每层之间是互通的，它们存在数据流的传递和交互，数据流可以跟踪到数据流向每层前后的变化，方便定位到是哪个逻辑层可能存在潜在 bug。

(4) 实例化定理库的构建

首先实例化定理库是定理实例化成的一个个的定理知识图谱，定理库是引擎的主要驱动依据，也是引擎匹配算法的对象之一，产生新知识和插入知识的来源之一。为了保证引擎的简单统一，实例化定理的书写需要满足一定的规范标准，在规范标准之内，引擎增加了一些预处理机制，增加了书写的灵活性。除此之外，在构建定理库时对实例化定理进行分类和标注，便于后续可以高效的匹配某一类实例化即可，降低引擎匹配的时间开销。

(5) 重构类人解答过程

在保证已经正确求解出答案和系统正确停机的基础上，引擎需要对推理的过程进行重现形成解答过程，这点类似人类的答题。需要将相关的推导过程展现出来，对于引擎就是将推理中使用的实例化定理以及计算中涉及的化简、解方程等技巧，用“因为 ***，所以 ***”的形式呈现。生成可读过程的方式有两种：后台日志的方式和知识图谱。解答过程的知识图谱需要注意的是，通过添加因果关系来表示前后逻辑。

1.4 本论文的组织结构

本论文共分为 6 个章节来详细介绍研究成果，具体论文的章节结构安排如下：

第一章：绪论。本章分成三个部分介绍初等数学类人解答系统的发展现状和背景，第一部分主要是介绍人工智能的发展历程以及不同发展阶段取得的重大研究成果，另外给出当前人工智能面临的一些难题；第二部分描述了自动推理的发展历程以及不同阶段的技术手段和算法思想；第三部分是对当前的“人工智能 + 教育”的现状进行研究和分析。

第二章：相关理论技术。首先，第一部分描述的是知识表示的几种不同方式以及知识图谱表示知识的特点。第二部分是详细介绍了产生式系统，包括基本原理，组成和实现，本文的推理引擎的设计就是基于产生式系统。第三部分介绍了推理引擎的相关算法以及图匹配相关算法。最后介绍了推理引擎中使用的符号计算服务平台 Maple。这些基础理论和技术是推理引擎设计思想的基石，也为推理引擎的设计实现提供了重要的技术支持。

第三章：复杂图推理中的知识表示。本章主要研究在复杂图推理中的知识表示，知识存储等问题。首先研究的问题是知识来源，这是知识表示的第一个环节，本文的知识分为两种知识：知识点知识和实例化定理知识。其中知识点知识来源于初高中教材中的数学名词、概念、以及一些新定义；实例化定理来源初高中教材的公理、公式、以及标准答案的解题解题技巧和公式变形。然后研究的是对于纯文本描述如何进行语义理解，转换成机器可以识别的语义知识，自然语言理解。利用自然语言理解的相关技术进行命名体识别和关系抽取。自然语言理解之后需要进行知识表征，本文知识表示选取的是知识图谱。第二部分介绍了实例化定理库的构建，以及实例化定理在推理引擎中的书写规范。最后详细描述实例化定理的置换原则。

第四章：图同构的数学推理引擎的设计和构建。本章主要研究基于图同构的图推理引擎逻辑架构和图匹配算法的设计，以及推理引擎如何将复杂逻辑推理与

符号计算推理相融合。推理引擎主要从三个方面详细介绍，第一层面是从逻辑推理方式，描述了几种不同逻辑推理的算法思想和引擎中”先逆后正“的推理方式；第二层面从推理引擎的基本原理和整体逻辑架构，宏观上讲述引擎的整体设计；第三层面是推理引擎每一逻辑层的具体模块实现，以及一些重要模块对应的算法设计；最后，对推理引擎的输出重构类人解答过程和生成结果过程知识图谱。

第五章：测试与分析。本章的测试分为两个模块：单例测试和批量测试。单例测试的主要内容包括两个部分：连通性测试和类人解答过程测试。连通性测试主要是检验前端自然语言模块和后端自动推理连通性，推理引擎不同逻辑层间的连通性，推理引擎与外部接口之间的连通性；类人解答过程测试包括能否输出所有涉及的定理和公式，输出的定理之间的逻辑关系是否正确。批量测试主要是检测推理引擎每个模块和核心匹配算法的稳定性，以及对于不同题型的兼容性。并对批量测试的结果进行了详细的统计和分析，从解题准确率、不同题型的兼容率、每题的平均解题时长等几个不同维度来评判推理引擎的性能和发现可能存在的问题，以便更好的提升推理引擎的解题能力和稳定性。

第六章：总结和展望。首先对本文中的研究内容和成果做出了总结，并结合测试结果和测试数据进行不同角度的考察和分析，找出了推理引擎存在的不足，依据不足提出相应的解决方案。对于一些疑难问题也给出了自己的思考。

第二章 相关技术和理论

近些年来，人工智能的发展非常迅速，经历了几个不同的阶段。从最开始计算智能到感知智能的阶段，正在迈向认知智能的阶段[16,17]。但要实现认知智能，第一步是如何让机器学会理解人类复杂语言中所包含的语义，在理解自然语言的语义基础上，进而对已有的知识做逻辑推理，无论是传统的规则演变还是深度学习的手段，如何将深度学习与逻辑推理相结合是人工智能当前所面临的难题。目前深度学习在自然语言处理中的命名体识别、关系抽取均取得不错的效果，知识图谱是一种表征自然语言语义的很好的载体，在知识图谱的基础上做知识演变、逻辑推理是当前研究的热门方向。

2.1 知识图谱

2.1.1 知识图谱概念

知识图谱这个概念最早是由谷歌在 2012 年正式被提出，当初提出知识图谱概念是为了优化谷歌搜索引擎的搜索效率，缩短响应时间。用图的形式存储知识，从而代替传统基于关键词的文本数据。事实上，知识图谱的起源可以追溯到更早的上世纪 60 年代的语义网络，经过长达半个世纪的发展和演变，才有了如今知识图谱完整的表示。1960 年 Quillian 首次提出表示知识的全新模式，即语义网络，用网络中的点表示概念、事物、对象等，网络中的边表示概念实体之间的关系。尽管这种知识表示模式很快得到大家的接受，非常直观简洁，但是很难应用到实际场景，主要原因在于网络中的节点和边完全用户自定义，过于随意，没有统一的规范和标准，因此很难去编码。

到 80 年代左右，相继出现本体论 (Ontology) 和万维网的概念。本体论可以理解成一种计算机模型，用于描述抽象实体或者概念、关系以及属性之间所组成的知识结构模型，基于此知识结构模型可以去进行知识推理。1989 年 Time Berners-Lee 发明了万维网，实现了文本间的链接。万维网通过超文本标记语言 (HTML) 将超文本链接从一个站点跳到另个站点[18]。1999 年万维网之父 Tim Berners-Lee 提出了语义网（Semantic Web）的概念[19,20]，语义网可以理解为在万维网的基础上对数据进行了粒度降解处理的一种网络流通的通用框架模型。将以往基于 document 的数据格式碎片化成更小的数据源 data，这些数据源 data 同样构成一个庞大的信息网络，因此如何去在不同的 data 之间信息传递和链接时语义网研究重点，即 linked Data，对于不同的信息源，将其从 page 或者 document 转换成 Data 的数据格

式，并将这些数据作为公开数据集，为了保证网络流中每个 data 的规范性和唯一性，通常参考已知本体进行建模并赋予 data 一个 URI 进行标识。

知识图谱在知识表示上具有天然优势，形式上以图的形式表示客观世界中的实体以及实体之间的关联，其中这些实体是由概念、事物、人、对象组成。这些实体和关系可以用最小的表示形式去建模，即三元组结构。三元组结构可以表示两个实体之间的关联，这两个实体可以是相同实体，也可以是不同实体。例如人属于灵长类生物，这句描述的三元组建模结构是 < 人，属于关系，灵长类生物 >。三元组在图谱中的呈现形式就是对应两个点，关系则是它们之间的边。相对于传统的语义网而言，它的数据粒度变得更小，更见直观的表示实体本身，这样完成了文档或者网页的超链接万维网到实体于实体之间关系互联的转变。这种转变不仅仅是数据粒度变得更小，知识图谱引擎除了提供类似传统的查询、搜索、字符集匹配等基本功能外，它包含了更深层次的语义知识。因为之前的文本首先会经过自然语言理解，抽取实体以及它们之间的关系，然后在利用知识图谱进行表征。因此用户进行人类语义查找时，其实会有自然语言理解之后对应的实体关系，知识图谱中显示出所有同义和关联的语义实体。因此，知识图谱积聚了传统知识搜索知识表示模型的优点，同时又具有自身特色，它庞大的网络关系图可以表示很复杂的语义知识，通过对网络中的实体和关系查询可以显示出它们的内在联系。

随着移动互联网技术的飞速发展，人们每天浏览新闻、社交网络、电子消费等会产生海量数据，如何快速的进行信息检索，精准的信息推送背后都必定有庞大的知识图谱作为支撑。金融知识图谱，医疗知识图谱，教育知识图谱均在行业内发挥着知识承载的作用，虽然知识图谱应用已经十分广泛，有些应用已经开始产品化，但不能忽视知识图谱带来的问题和面临的挑战。知识图谱目前面临的挑战包括三个方面。第一方面，是知识图谱构建的问题。不同领域知识图谱对构建的标准很高，只有行业内专家才能构建出准确、完整的知识图谱，但是缺点也是显而易见的，人工成本过高，而且耗时效率低下。所以如何自动或者半自动构建知识图谱成为研究重点。第二方面，知识图谱与深度学习相融合。对于急剧增加的海量数据，传统搜索图搜索算法效率会下降明显，需要利用深度学习去处理图数据。图嵌入、图编码、图匹配等一系列问题需要考虑。第三方面，图上的逻辑推理。知识图谱与逻辑推理结合是目前的难点，如何在图的节点之间体现逻辑关系，以及图节点如何引入计算都还处于研究阶段。

2.1.2 知识图谱模型

知识图谱的构成主要指从数据模型的角度分析，它包含数据层 (data layer) 和模式层 (schema layer) 两个部分。其中，模式层是知识图谱底层的概念存储，是数据模型的核心组成部分[21,22]。主要是本体概念和概念之间的关系构成的。这些概念实体可以理解成对上层数据层的具体实例化实体和关系一种规范化约束，因此数据层与模式层是一种依赖约束关系。模式层约束着数据层，而数据层由同样看作模式层本体的实例化结果。在一些只承担搜索查询的知识图谱中，通常只需构建数据层节点和关系即可。模式层在推理知识图谱中有着重要应用，它可以从本体的约束关系中构建相对应的抽象实体类和抽象关系类，同时模式层采用的树状类结构，类之间的继承关系可以在 Java 中类继承保持一致性，方便编码工作。关系是有向性，有向性分为单向有向性和双向有向性。关系中可以存储属性，同样关系也可以有继承结构。

在数据层，是具体模式层的本体和关系的实例。对于每一组的实体和关系实例，可以用三元组结构 < 实体-关系-实体 > 去描述。所有的三元组结构组织成了关系图，值得注意的是，三元组中的实体和关系还可以存储属性，例如数学表述“点 P 到直线 l 的距离为 d”，解析成的三元组结果为 < 点，距离关系，直线 >，而“距离”作为距离关系的属性，属性值为 d，属性无论是在实体或者关系中的存储都是 key-value 键值对的方式。对于属性的存储与三元组存储可以相互转换，上述例子中“距离”可以单独视为一个实体-距离实体，这样便完成 < 实体-属性-属性值 > 到 < 实体-关系-实体 > 的转变。

如图 2-2 所示是知识图谱数据模型的构成图实例。

2.1.3 知识图谱构建

通过上一章节对知识图谱的构成介绍可知，知识图谱的数据模式包含模式层和数据层，二者的依赖关系是模式层是顶层架构，而数据层是底层应用，是模式层的具体实现。从构建的方向可以分为两种构建方式，自底向上和自顶向下[23,24]。自底向上的构建方式是先从不同的数据源中抽取实体、关系、属性等知识，然后将知识添加到知识图谱的数据层形成三元组结构[25,26]，当形成的三元组知识逐渐丰富时，对知识图谱中的实体关系进行分类、归纳、整合处理，抽象成顶层的本体和关系结构。因此自底向上是先数据层后模式层构建的方式。自底向上的应用场景主要是大型的共有库构建，因为大型共有库构建是所有人都可以参与，知识来源形式多样、知识表述多元化, 这样的知识库很难在最初的时候确定模式层的本体和关系结构，只有通过不断的累积数据然后再去抽象数据共性成概念本体。自

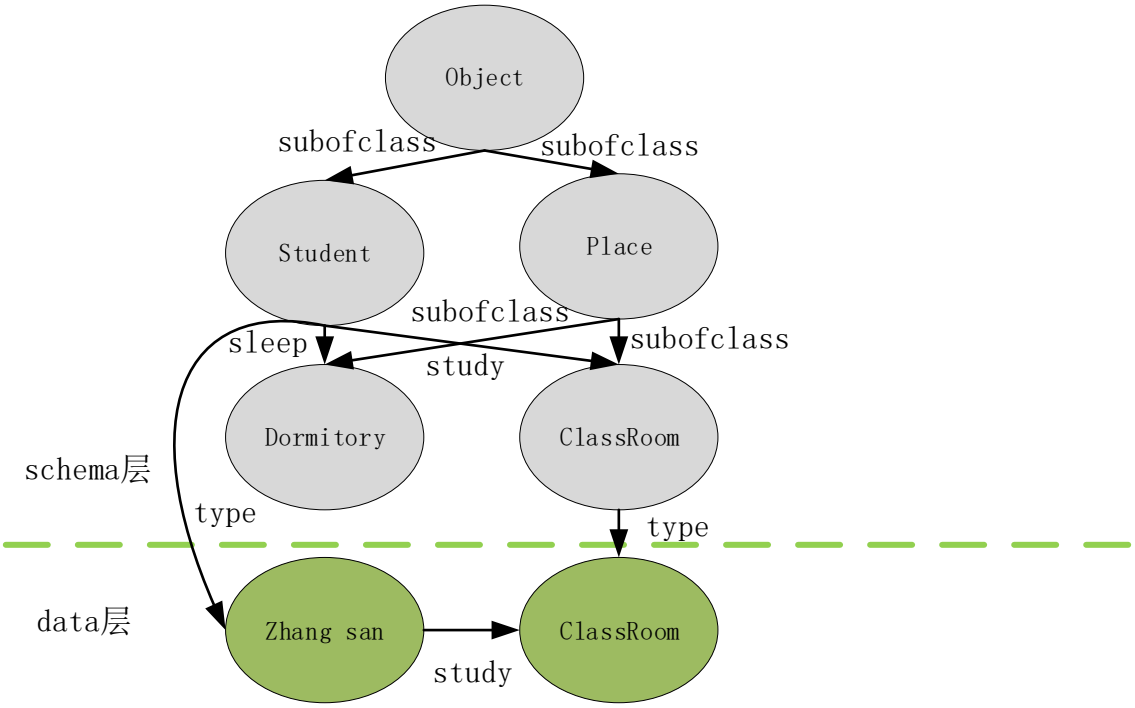


图 2-1 知识图谱构成图实例

顶向下则是相反的构建思路，先构建好所有本体以及本体间的关联，然后再根据数据源去构建数据层，在顶层约束下对数据层进行规范化、标准化构建本体实例和关系。这种构建模式一般应用于行业内的知识图谱，知识图谱的规模有限，构建的知识图谱质量高，但缺陷在于需要专业人士才能构建并且使用范围有限。因此，当前正在积极探索将二者相融合的构建方式，发挥各自的优点。

在明确知识图谱的构建方式后，数据来源和具体构建流程问题是构建的重点。数据来源是构建知识图谱的首要问题，不同的数据源会有不同的数据结构，因此需要对数据进行分类处理，不同类采用不同的构架方式。通常情况下，按照数据结构化程度将数据分为三种：结构化数据、半结构化数据和非结构化数据[27]。结构化数据具有数据结构清晰、界限明确、格式规整的特征，多为关系型数据库中存储的数据[28]，对于结构化数据通过 D2R 技术转换成 RDF(linked data) 格式的数据，D2R 是由 D2R Server，D2RQ Engine 和 D2RRQ Mapping 语言三个部分组成。D2R Server 可通过 HTTP 请求提供外部查询的接口访问服务，主要是可供浏览器调用或客户端调用。D2RQ Engine 主要功能是将 D2RRQ Mapping 文件转换成 RDF 格式[29,30]，是基于 jena(一种语义网的 Java 平台) 的接口。D2RRQ Mapping 是将关系型数据的结构化数据虚拟映射成 RDF 数据格式的 Mapping 规则集。相对于结构

化数据，半结构化数据来源更加广泛，例如网页中的数据、百科的数据，这些数据都具有有相对结构的数据，但又不能直接使用需要进一步处理。对于半结构化数据，可以使用传统的正则表达式去匹配然后转换成结构化的数据，再利用结构化数据转 RDF 格式的相关技术处理，当数据集有一定规模时，可以采用有监督学习的方法在有标注的训练集上学习抽取规则，再去新的网页数据抽取。实际情况最普遍的数据格式是非结构化的，人类的语言就是典型的非结构化数据。处理非结构数据的第一步需要借助自然语言理解，命名体识别和关系抽取，将非结构化数据的语义特征转变成三元组结构。

知识图谱具体构建流程由四个部分组成：原始数据、知识抽取、知识修正、数据模型规范。知识修正主要处理同一实体多个类型以及隐含实体指代消解问题，是自然语言理解的语义理解修正。数据模型规范是构建知识图谱的模式层，顶层设计来约束底层的数据层数据标准化。如图 2-3 是知识图谱的构建流程图。

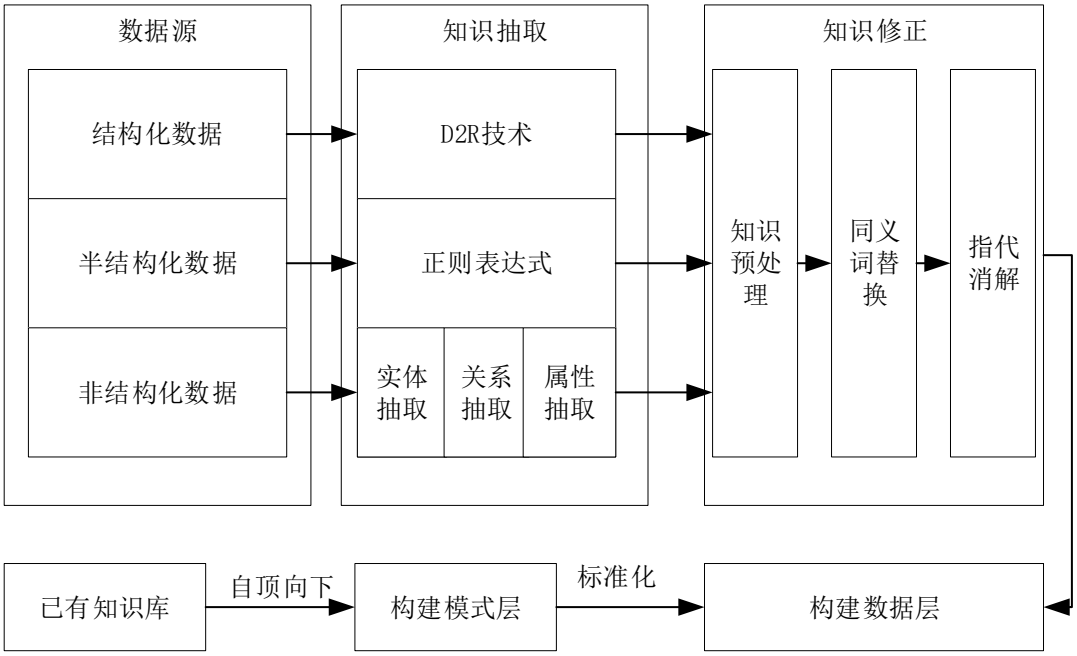


图 2-2 知识图谱构建流程图

2.2 图数据库

2.2.1 图数据库概述

在完成原始数据到知识图谱构建之后，需要考虑以什么方式去存储知识。一种是基于关系型数据库进行存储，一种是基于图数据库存储。关系型数据存储是

将知识图谱转换成 RDF 三元组 <Entity,Relation,Entity> 进行存储[31]，RDF 存储的优势在于利于共享和发布数据，但 RDF 三元组存储时会丢失属性信息。基于图数据库存储是以图作为基本模型，图中的节点和边分别存储实体和关系信息，同时图数据中的节点和边都可以存储其属性信息，图数据库在查询和搜索效率是高效的，也可以表示复杂的语义知识，对知识图谱的显示作用也更好，其次，传统的关系型数据是以表的形式存储数据，对查询关联数据需要不断联表查询，效率较为低下，而图数据库对关联查询友好支持，整体来说更适合作为知识图谱的存储。如表 2-1 所示图数据库与其他类型数据库的对比。 图数据库近些年来之所以得到

表 2-1 不同类型数据库对比

数据库类别	数据存储结构	特点	实例
图数据库	图	知识表示间接直观，关系查询高效	Neo4j
关系型数据库	数据表	数据以行和列的方式存储在表中，支持联表查询	Mysql
键值数据库	哈希表	一般直接存储在内存，查找速度快	Redis
文档数据库	键值对扩展	非结构化直接对文档插入数组数据	MongoDB

广泛的认可，最主要的原因在于它在不同的应用场景中相对于传统的数据库在知识表示、搜索查询、关联查询都有明显的效果提升。现在越来越多的领域都在构建知识图谱，搜索领域，谷歌最早就用知识图谱做搜索优化，后面国内的百度百科也开始构建共有知识库；社交领域，FaceBook，Twiiter 等社交平台利用知识图谱进行好友推荐，新闻推荐；零售领域，ebay，沃尔玛使用它进行商品的实时化和个性化推荐，提升用户购买体验。另外，图数据因为其表示形式简洁直观也是受到青睐的原因之一。它利用图数据结构来存储客观时间的语义知识，图可以看作是点集和边集的组合，其中节点是图数据库主要数据元素，边是连接不同节点之间的关系。

2.2.2 图数据库 Neo4j

当前市面的图形数据库百花齐放，目前 Neo4j 以绝对的优势成为最受欢迎的图数据库。Neo4J 是 Neo Technology 所提供的基于 Java 实现的开源图形数据库[32]。Neo4J 与其他数据库一样支持事物以及事物的 ACID 四大特性，同时 Neo4J 的专业版支持集群和主从复制。Neo4J 之所以能够表示复杂语义，主要是依赖其图模型。图模型从标签、节点、关系、属性四个方面入手。标签是对节点和关系作为分组的一种标识，可以提高查询的速度和分类管理。一个节点和一条关系可以拥有一个标签，也可以是多个标签，依据标签索引便可得到同标签的所有节点和关系；

节点存储知识抽取中的概念实体，Noe4J 的节点支持属性以 key-value 键值对存储，节点中的属性可以是一个或者多个；边存储的是实体之间的关系，表示方式是有向边，与实体一样关系也支持属性存储；对于属性的存储 Neo4j 只支持键值对存储，属性可以被索引和查询。Neo4J 目前支持的数据格式有整形常量，字符串以及字符串数组，不支持对象存储，当属性是对象时就会出现无法兼容的情况。

Neo4J 作为一种图数据库有自己的查询语言和语法，Cypher 是 Neo4J 官方提供的图形查询语言，单个 Neo4J 实例可以存储几十亿的节点和关系数据，它的查询功能非常强大，是汲取各种数据库查询语言的思想并应用在图上查询。Cypher 是一个申明式的语言[33]。Cypher 查询语言常用关键字“match”、“create”、“where”、“filter”、“project”等用于条件过滤查询，Neo4J 同时提供了相关的操作函数，可用于节点查询、字符串处理等。

Neo4J 为了支持更强大的图查询和图重构功能，它还支持各种插件功能，APOC 是 Neo4J 后续版本中提供的过程调用包，这些过程调用函数是 Java 实现的，可以直接部署到 Neo4J 实例里面包含丰富的过程调用和复杂图查询算法，如下表 2-4 所示是 APOC 常见图算法过程调用。

表 2-2 部分 APOC 过程调用

过程调用	参数	功能描述
apoc.path.expand()	startNode <id>|Node, relationshipFilter, labelFilter, minLevel, maxLevel	路径扩展
apoc.path.subgraphNodes()	startNode <id>Node/list, maxLevel, relationshipFilter, labelFilter, bfs:true,limit:-1, optional:false	在子图中扩开节点
apoc.path.subgraphAll()	startNode <id>Node/list, maxLevel, relationshipFilter, labelFilter, bfs:true, filterStartNode:true, limit:-1	扩展子图 并返回所有子图路径
apoc.algo.dijkstra()	startNode, endNode, ’KNOWS|IS_MANAGER_OF>’, ’distance’	dijkstra 算法搜索最短路径算法
apoc.algo.aStar()	startNode, endNode, ’KNOWS|IS_MANAGER_OF>’, ’distance’,’lat’,’lon’	A* 算法搜索最短路径

2.3 自动推理

2.3.1 产生式系统

产生式系统是人工智能认知模型中最典型的体系结构之一，其最核心的思想通过依据产生式规则对已知事实进行推理演变，从而产生更多的知识，将新生成

的知识重新加入到已知事实库中，然后继续迭代。从核心思想可以总结出产生式系统的逻辑结构，首先是已知事实，由于已知事实的输入需要转化成计算机能够理解的知识，便涉及到知识表示问题；其次是产生式规则，产生式规则的基本形式是“<If P，then Q>”[34]，其中 P 代表规则的前提条件集合，Q 的含义分为两种：动作或者结论。当 Q 表示动作时，意味着当满足 P 条件时，规则会触发相对应的动作；当 Q 代表结论，则在满足所有前提条件时，会生成结论知识。最后是产生式系统的控制策略，控制策略负责对整个系统进行流程控制、策略调度、规则选取及应用、动作执行等。如图 2-4 是产生式系统模块图。 逻辑推理的三种推理方

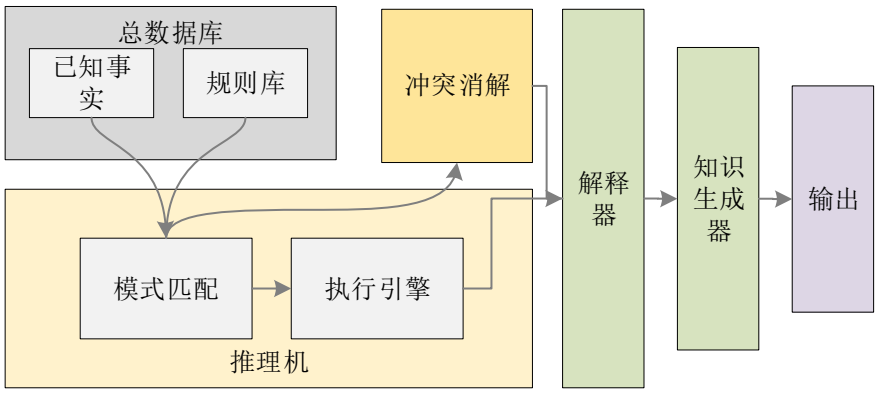


图 2-3 产生式系统结构图

式：正向推理、逆向推理和正逆结合，均可适用于产生式系统。具体工程中会根据实际情形选取不同的产生式系统，正向推理的产生系统是一种暴力搜索思想，适用规则库规模中等的系统，若规则库规模过大，会产生知识爆炸问题，下面详细阐述正向推理的算法过程：

1) 首先将已知事实经过知识抽取，自然语言理解成计算机表示的知识，全部放入知识库中。

2) 读取知识库中已知事实，将已知事实”(conditions,askings)” 划分成前提条件 C 和求解目标 A 两个部分。

3) 在规则库选取规则，将产生式规则“<If P，then Q>”划分成前提条件 P 和结论或者动作 Q 两个部分。

4) 将步骤 2) 中已知事实前提条件 C 与步骤 3) 中产生式规则 P 进行匹配，若匹配成功，执行步骤 5), 若匹配失败，回溯到算法的步骤 3）执行。

5) 将产生的新知识与步骤 2) 中已知事实求解目标 A 进行匹配，若匹配成功，则结束正向推理，推出；若匹配失败，执行步骤 6)。

6) 将新知识插入到已知事实库中，若插入成功，回溯到算法的步骤 3）执行；若插入失败，进入知识冲突消解策略。

与正向推理的盲目式搜索不同，逆向推理的前提是已知事实中求解目标明确，逆向推理的核心思想就是分解子目标，然后再去不断规约子目标，直到到达所有满足条件的事实，终止规约。下面详细阐述逆向推理的算法过程：

1) 首先将已知事实经过知识抽取，自然语言理解成计算机表示的知识，全部放入知识库中。

2) 将已知事实中的求解目标分离，构建规约树的根节点。

3) 利用分支技术分解求解目标节点，生成多个子目标。

4) 去规则库中寻找子目标对应的产生式规则，若能够找到对应规则，执行步骤 5); 若找寻失败，回溯到算法步骤 3)。

5) 将产生是规则的前提条件分解出来，若所有的前提条件都存在已知事实，停止规约，执行步骤 6)；若存在某个前提条件不存在已知事实，继续规约，回溯到算法步骤 3)。

6) 遍历规则的规约树，生成规则路径。

从上文的正向、逆向两种算法可以看出，无论哪种推理方式，都需要规则库的支撑。尽管产生式系统逻辑结构清晰，模块化独立程度高，也适用多种推理方式，但也有自身的缺陷和不足。最主要的问题在于当规则库规模过大时，系统缺陷就会比较明显，规则数量多是会对匹配的精度要求高，每一个规则都需要经历同样的推理流程，时间消耗较大，产生的知识越多，继续迭代会产生知识爆炸，同时知识冲突的概率也会急剧增加。因此针对这些问题，需要引入知识减枝的策略和知识冲突消解的策略。

2.4 匹配算法

基于规则的推理引擎，其核心算法思想就是匹配算法，通常情形下，匹配算法是对已知事实与待匹配事实搜寻验证的过程。由于不同规则库知识表示的有所差异，决定了匹配算法设计是不同的，主要根据产生式规则和实例化知识图谱规则将匹配算法分为两大类：模式匹配和图匹配。

2.4.1 模式匹配算法

模式匹配是指将已知事实与知识库进行匹配，从匹配对象类型可以分为字符串匹配，类对象匹配，或者规则匹配，三种模式匹配的主要区别在于匹配粒度，其中规则匹配粒度最粗，字符串的匹配粒度最细。另外类对象匹配和规则匹配依赖

于字符串匹配，本文中引擎的匹配算法是融合三种模式匹配方式，能够处理较为复杂的匹配搜索问题。规则匹配相对于字符串匹配较为复杂，首先定义规则条件部分 (left-hand side), 简称 LHS，结论部分 (right-hand side)，简称 RHS。规则分为条件和结论两部分，条件部分参与事实匹配，结论部分可以产生新知识，而事实集满足匹配条件的前提是包含所有规则集条件。具体的算法流程如下：

1) 将选取的实例化规则 R，构建条件集 LHS 和结论集 RHS。

2) 从已知事实集中构建出等价于条件集 LHS 同等大小的事实子集 C。

3) 若已知事实集 C 满足 LHS(R)=true，则将已知事实子集 C 和对应规则 R 加入到冲突集；否则，执行步骤 4)。

4) 对于 M 个事实选取别的组合构成新的事实子集 C1，若没有新组合，执行步骤 5); 否则，回溯到算法的步骤 3)。

5) 去规则库选取新的规则 R1，执行步骤 2)。

关于模式匹配算法最为常见的有 Rete 算法、Tread 算法、Leaps 算法，这三种算法都是前向规则快速匹配算法，递进式推理的特点是所有的动作都是串行执行的，匹配动作是最先完成的，依据匹配动作的结果去执行实例化规则的选取动作，然后再去不断去循环之前的动作。其中，rete 算法应用最为广泛，很多规则推理引擎都直接借鉴于其中的思想。Rete 算法包含两个部分，规则编译和构建 rete 网络，已知事实可以在 rete 网络图中流动。rete 具体算法过程如下：

1) 将已知事实 F 加入到工作内存 working memory

2) 将已知事实 F 于规则库选取的规则 R 进行模式匹配，若匹配成功，将事实加入到匹配元素表；否则执行步骤 3)

3) 激活多个规则，并将规则放入到冲突集中

4) 解决冲突，将冲突的规则按照先后顺序放入到 Agenda

5) 引擎去执行 Agenda 中的规则，重复步骤 2-5)，直到所有 Agenda 所有规则执行完毕。

2.4.2 图匹配算法

图匹配是个经典的图上匹配问题，图匹配问题涉及二分图匹配、最大连通域匹配、子图匹配等问题。本文设计的图匹配引擎采用的算法思想就是子图匹配算法，又称子图同构算法。所谓子图同构任务，即给定一个 target graph A（m,n）, 和一个 query graph B(p,q)。其中，m、p 是点集，n、q 是边集。试图寻找一种映射关系，使得两个图对应的点的节点类型相同，query graph 中的边在 target graph 中对应的点之间都存在，节点名称可以不同，且节点类型相同。如下图 3-4 所示，两个

实例化子图是同构图：**********

本文设计的图匹配算法是依赖于知识图谱实现的，是三元组匹配和图路径匹配融合算法。三元组匹配首先将图拆解成＜实体，关系，实体＞的三元组形式，三元组中的实体和关系是对象匹配，只匹配类型是否一致，实体和关系类型全都匹配上，三元组匹配成功。由于三元组匹配会丢失图中路径信息，因此路径匹配是对三元组匹配的检验，路径匹配是从三元组中的节点出发，搜寻从该节点出发的所有关联节点，包括深度有限遍历和广度优先遍历。检测关联节点集是否包含关系，若是包含关系，可以认为三元组严格匹配。具体的图匹配算法如下：

1) 读取实例化定理图和题目知识图谱

2) 将实例化图和题目知识图谱结构成三元组集合 QTripleSet,RTripleSet

3) 执行三元组匹配，若匹配成功，执行算法 4); 否则回到步骤 2)

4) 执行路径匹配算法，若匹配成功，回溯到算法 2)，直到所有规则三元组集遍历结束；否则推出匹配

5) 步骤 4) 的所有三元组匹配均成功，子图匹配成功。

2.5 符号计算平台

本论文研究的课题是初等数学问题求解，初等数学中涉及到大量的数学符号计算，因此推理引擎的推理方式中融合了计算推理。计算推理主要来源于计算机代数系统的支撑，目前计算机代数系统的平台实现有 Matlab、Maple、Mathematica 以及基于 python 的 sympy 等。Matlab 是偏向于数学图形的绘制，在符号计算方面表现相对于其他代数系统较弱；Mathematica 是当前最强的浮点数计算平台，但由于初等数学中涉及的主要是符号计算，并且 Mathematica 只提供 C 编程语言的接口，而引擎的构建使用的是 Java 语言，出现接口不兼容的情况；结合实际的研究问题，最终选用 Maple 符号计算平台，既能支撑初等数学中符号计算需求和图形绘制，同时 Maple 底层是基于 Java 虚拟机实现的，很好的兼容 Java 项目接口。Maple 提供了丰富的计算函数库，例如化简”simplify “、解方程”solve “、恒等式”identity “、半代数集”SemiAlgebraic “，分解因式”factor “，这些基础计算接口都在推理中有所应用。以上的计算函数命令集成了高等数学算法，可以应用于初等数学的解题。但同时 Maple 符号计算平台也会存在一些问题，Maple 平台在计算是不支持多线程，这对并发编程支持不友好，除此之外，Maple 计算函数集成的高等数学算法，可能会导致类人解答过程的信息丢失。

2.6 本章小结

本章主要介绍了知识图谱、产生式系统、匹配算法和符号计算平台四个方面的内容。首先对知识图谱进行发展历程以及不同时期的特点进行了回顾，然后阐述了知识图谱的应用以及面临的挑战，介绍完知识图谱的概况后重点介绍知识图谱的数据模型结构、知识图谱构建以及存储等问题。第二方面是介绍了本文中使用的系统模型产生式系统，并给出了不同推理方向的产生式系统的详细算法。第三方面介绍了推理引擎中的核心算法，从模式匹配入手，讲述了几种常见匹配算法。最后是介绍计算推理中使用的符号计算平台 Maple 的基本功能，同时对比了不同数学软件之间的区别。

第三章 复杂图推理中的知识表示

3.1 概况

本章主要是介绍图同构推理引擎在复杂图推理中的知识表示问题，主要分为两大模块，一个是初等数学领域概念知识图谱的构建，另一方面是实例化定理库的构建。其中概念知识图谱的构建从知识图谱构成、知识图谱存储、知识图谱的应用三个方面介绍；实例化定理库的构建则从定理来源、定理库创建、定理书写规范、定理的应用四个方面介绍。总体关系图如图 3-1 所示。

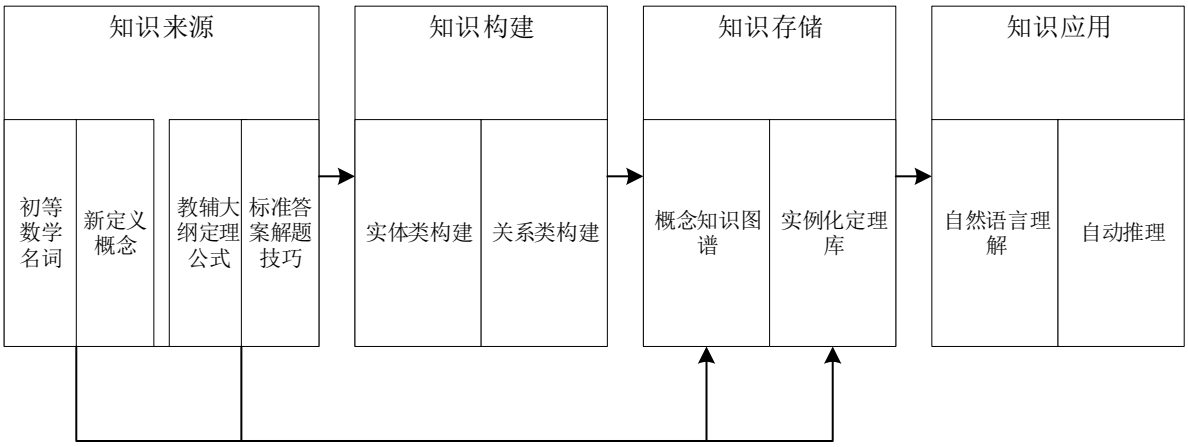


图 3-1 知识表示关系图

3.2 概念知识图谱的构建

3.2.1 知识图谱构成

知识图谱是用一个逻辑上的图去表征语义，这个图可以是连通的，也可以是非连通的。本文中构建的初等数学领域的概念知识图谱就是有一些离散图组成的逻辑上的大图。无论是离散图还是逻辑上的图，它们的最小组成单位都是三元组结构，本文中设计的图同构推理引擎的最小推理结构就是知识图谱中的三元组。三元组则是由两个实体和它们之间的关系构成的，实体和关系又是三元组的最小组成单位。例如，”三角形 ABC 与三角形 DEF 相似“，解析成三元组结构为（三角形, 相似关系，三角形）, 这里的头尾实体均为”三角形“，关系为”相似关系“。这里为了描述方便起见，以实体和关系的类型字段表示，但对于后续推理来说，实

体类和关系类的字段和属性信息远丰富的多。接下来详细介绍本文如何构建实体类和关系类。

（1）实体类的构成本文中实体都是来源于初等数学中基本概念、名词，如”函数“实体、”向量“实体、”数列“实体、”双曲线“实体等。当前构建的实体类共236个，为了保证实体类的可扩展性和灵活性，在最初的架构设计上采用了”开闭原则“和应用 Java 语言的继承、多态以及反射的思想，即我们构建出一个抽象实体类 AbstractData, 在 AbstractData 中添加共有的属性字段，以及相关的函数方法，再让所有的具体实体类继承 AbstractData，这样的好处有两点：第一点可以做到很好的扩展性，可以随时添加新的实体类；第二点方便编码，当不知道是哪个实体类时，在编译时期可以直接创建抽象父类实体类，再利用反射技术指向不同的子类对象。

抽象实体类除了上文中描述的作用，它的一些字段在推理中也至关重要。如表 3-1 是 AbstractData 抽象实体类的具体属性字段表。

表 3-1 抽象实体的属性表

属性名	中文含义	描述
EntryName	实体名	用于命名实体，在知识图谱和类人解答过程中显示
Id	实体的唯一标识	来唯一标识实体
TypeName	实体类型	实体的类型应用于三元组匹配
Knowledge	新知识	存储知识更新阶段的新知识
Used	是否使用过	用于记录实体在遍历时是否已经出现，避免重复

（2）关系类的构成关系是表示两个实体之间的关联，因此关系类中首先会记录它所关联的头尾实体的信息，这里为了区分关系名和实体名，自然语言处理中会统一给抽取的关系命名加上”Relation“后缀, 如首项关系”FirstItem Relation”。为了保证图推理引擎的匹配精确度，关系类构建的需要符合以下几条准则：

1）关系的命名尽量规范，头尾实体名加上“Relation” 后缀；

2）关系名的唯一性，若出现关系名重复，代码运行时会有反射异常；

3）关系构建尽量精细化，如”点在直线上“,”点在圆上“，我们应当构建”PointOnLineRelation”,”PointOnCircleRelation”, 而不是都构建成“OnRelation”。

同样的，与实体类建模相同的是，所有的具体关系类也都继承于同一个抽象父类 AbstractRelation。与 AbstractData 一样，AbstractRelation 也有关系特有的一些属性值，不同的属性值在推理中发挥着不同的用途。下表 3-3 是 AbstractRelation 抽象实体类的具体属性字段表。

表 3-2 抽象关系的属性表

属性名	中文含义	描述	数据类型
RelationName	关系名	用于命名关系，在知识图谱和类人解答过程中显示	String
RelationId	关系的唯一标识	来唯一标识关系	String
IsSelfCircle	是否自环关系	应用于自环映射判断	boolean
FullName	关系全路径名	通过全路径名来发射到相应的具体关系类	String
Order	关系序	用于对相同关系的三元组重排序	Integer
IsSameGroupe	是否同组	用于消除假组合降低组合次数	Integer
RelProperties	关系属性	用于支持属性匹配和更新	Map
IsConclusion	是否结论三元组	用来标记三元组类型，区分条件三元组和结论三元组	Integer
CreateBranch	创建分支	判断该关系是否触发了分支	boolean
Knowledge	新知识	新知识为关系的属性	String

3.2.2 知识图谱存储

概念知识图谱就是由这些实体类和关系类通过转换成图谱中的节点和关系，然后存储在 Neo4j 图数据库中。Neo4j 是一种图数据库，图中节点可以存储实体类相关信息，Neo4j 支持的数据格式是字符串以及字符串数组，对于每一对属性值是 k-v 键值对存储的形式。同样的，图中节点之间的边就是用来存储关系类的相关信息。这里特别说明，由于 Neo4j 图数据库数据存储不支持对象存储，这对后面有些属性作为对象存储并不是友好兼容，例如，“两直线相交于点 P”这是三元关系的描述，如何在知识图谱中表征三元关系，有两种处理方式。第一种解构成一个三元组（直线，相交关系，直线），其中实体点 P 作为相交关系的属性存储；第二种是将三元关系转换成多组两元关系，（直线，相交关系，直线），（点，在直线上关系，直线），（点，在直线上关系，直线），这样便实现了多元关系的存储。在后续的推理中我们大多数使用第二方式去存储多元关系。如图 3-2(a) 和如图 3-2(b) 分别是三元关系的知识图谱两种不同存储示意图。**********

3.2.3 知识图谱的应用

本文中初等数学知识图谱主要应用在自然语言理解的关系抽取中和后端解题的实例化子图。自然语言理解是整个解题系统求解过程的第一步，后端的推理也是依赖于前端自然语言理解的结果，因此要求自然语言理解要尽可能地命名体实别准确以及保证这些实体之间的关系抽取的准确度。初等数学概念知识图谱便给关系抽取提供一定的支撑，首先命名体识别是自然语言理解的第一步，命名体识别出来的实体则对应于知识图谱中的实体节点。自然语言理解和知识图谱是个相

辅相成、相互完善的过程：当出现自然语言理解识别出的实体在知识图谱中没有，需要在知识图谱中补充；当自然语言理解识别的实体于知识图谱中的不一致，需要对自然语言理解实体命名做修正。进一步，对于关系抽取的第一步，输入任意两个实体，去概念知识图谱中反查出这两个实体所包含的所有关系，最终抽取的关系必须在这组关系集合中，因此知识图谱的构建对自然语言理解至关重要。

后端解题中同样需要依赖于知识图谱。本文构建的是一个图推理引擎，推理引擎的入口就是自然语言理解实例化出来的子图，其中包括题目子图和规则子图。无论是题目子图和实例化子图都必须能在概念图谱中包含其所有的实体和关系，如果在概念图谱中找不到子图对应的实体或者关系，推理引擎则会抛出反射异常，需要自然语言理解或者知识图谱做相应的修正和补充。

3.3 实例化定理库的构建

3.3.1 定理来源

本文要构建的是初等数学领域的实例化定理库，定理的范围是初等数学领域，定理的来源于三个部分：第一部分是目前初中、高中的数学教材上的定理以及公理，同时参考市面上主流教辅书籍（如《高中数学知识清单》【】）和互联网上相关的信息；第二部分是从标准答案的解题过程中抽取相应的解题方法、解题技巧，为了保证自然语言理解的可靠性，标准答案会做些微调，尽量保证标准答案中描述的实体类型明确，可以是一个短句作为一个实例化定理，也可以是多个短句组合成一个实例化定理。第三部分是表达式变形，表达式的恒等变形，合并同类项，拆分移项等一系列操作转化成实例化定理。

为了方便定理库的管理和提升推理引擎对定理库的搜索效率，对定理进行分类收集、分类构建、分类存储。目前基于初等数学题型特征，将定理分成五个大类：函数、数列、向量、解析几何、平面几何。下表 3-4(a)、3-4(b)、3-4(c)、3-4(d)、3-4(e) 分别是五个类型定理公式表。

3.3.2 定理库创建

每条实例化定理创建是将上述定理公式经过自然语言理解生成三元组结构的 json 字符串，后端接受到前端实例化定理的 json 会生成定理知识图并存储在 neo4j 图数据库中，定理库则是将所有搜集的定理公式均转成实例子图存储在数据库中。目前实例化库的实例化总数为 323 条，其中函数 107 条，数列 89 条，向量 38 条，解析几何 101 条，复数 21 条。

实例化定理文本和题目文本在自然语言输入的时候是有所区别的。自然语言

表 3-3 函数实例化定理

定理标签	定理名称	描述
rKwkWrig	函数与坐标轴相交	函数 $f(x)=g(x)$ 与 x 轴只有 n 个公共点， 则 $\#publicPointOfFunction\#f(x)=g(x)\&n$
qyZkFwKa	函数到曲线	已知函数 F 的曲线为 E， 则曲线 $\#Func2Curve\#F$
XkylPYoG	函数单调区间	已知函数 $f(x)=g(x)$， 则 $f(x)=g(x)$ 的单调区间为 $\#GetFuncMonoItv\#f(x)=g(x)$
oOakjOWd	函数有零点	若函数 $f(x)$ 有零点 rsa， 则不等式 $\#haveZeroOfFunction\#f(x)$
IOekuEOt	函数求导	已知函数 $f(x)=g(x)$， $f(x)=g(x)$ 的导函数为 $\#diffOfFun\#f(x)=g(x)$
zsBlBQjc	求函数周期	已知函数 $f(x)$， 则 $f(x)$ 的最小正周期为 $\#CalPeriod\#f(x)$
JCekuEOt	求反函数	已知函数 $f(x)=g(x)$， $f(x)=g(x)$ 的反函数为 $\#inverse\#f(x)=g(x)$
UQbkuGfC	计算函数最大值	已知函数 $f(x)=g(x)$， 则 $f(x)$ 的最大值为 $\#CalculateMaxValueOfFunc\#f(x)$
DegkuGfC	计算函数最小值	已知函数 $f(x)=g(x)$， 则 $f(x)$ 的最小值为 $\#CalculateMinValueOfFunc\#f(x)$

提供”type“的属性字段，当”type“输入为 0 时，表示输入的时实例化定理。定理的文本输入分成两个部分，前提条件和结论分开输入，前提条件三元组和结论三元组是通过关系中属性字段”isConclusion” 区分，当”isConclusion” 为 2 时是结论三元组，其他取值为前提条件三元组。除了”type“字段，实例化定理还包含另外两个个标注字段：”instantiatedLevel “、”instantiatedCategory “、”instantiatedDescription “。其中，”instantiatedLevel “字段含义代表该实例化定理的优先级，范围值为 0-1000，0 代表优先级最高，数值越大优先级越低。”instantiatedCategory “表示实例化定理类别，范围值为 1-8，1 代表函数，2 表示向量，3 表示数列，4 表示解析几何，5 表示复数，6 表示平面几何，7 表示立体几何，8 表示表达式变换。通过自标注实例化属性字段，更加丰富了实例化定理的内容，并具体应用后续的自动推理和知识点标注中。实例化定理的详细标注字段如表 3-5 所示。

3.3.3 定理书写规范

上文中介绍了实例化定理的来源，为了保证推理引擎的简单性和统一性，需要对实例化定理进行规范化和标准化书写。首先，实例化定理是分成两个部分，前提条件和结论。其中，前提条件就是原始的定理文本描述，是实例化定理参与匹配的部分；结论部分是需要部分人工参与书写，它是推理引擎产生知识的依据，即当该实例化匹配成功时，推理引擎会解析定理结论部分，然后触发相应的符号计

表 3-4 实例化定理属性字段

属性字段	描述	用途
type	输入类型	用于区分实例化定理和题目输入
instantiatedLevel	实例化优先级	推理引擎的实例化选取
instantiatedCategory	实例化类别	实例化分类器
instantiatedDescription	实例化描述	类人解答显示和知识图谱显示
instantiatedKnowledge	知识点	记录推理路径中使用的知识点
instantiatedKnowledgeWeightsedge	知识点权重	知识点的优先级
IsConclusion	是否结论三元组	区分结论三元组和条件三元组

算服务，产生定理知识并依据不同的知识更新策略，将新知识插入到原有的知识库中，进行下一轮的知识迭代。因此推理引擎对实例化结论部分的书写提出了以下准则：

(1) 标准化实例化定理结论部分由三个部分组成：操作函数、定理公式、形式化参数。操作函数是触发定理产生新知识的一组动作，操作的对象就是定理公式以及形式参数置换后的具体参数，在书写时使用“#”将操作函数进行区分，操作函数有时可以省略，当省略时，引擎会默认添加“simp”操作函数表示化简操作；定义公式是操作函数操作的对象之一，它可以是书中的定理公式、等价变形或者表达式变换，例如求等差数列前 n 项的通项公式“$a_n = a_1 + (n-1) * d$”,”$x^2 + 2 * x * y + y^2 = (x+y)^2$”等都属于定理公式的范畴；形式化参数指的是定理中抽象化参数，不参与具体计算，需要注意的是定理公式和形式化参数之间是”:“隔开，而参数于参数之间是”&“符号隔开。最终，完整的一条实例化定理的结论部分便书写完成。如求等差数列通项公式的实例化定理结论为：”#simp#$a_n = a_1 + (n-1) * d$：a_1&d “

(2) 如表 3-3 列举了常见的操作函数表

(3) 上文中介绍过形式化参数是不直接参与计算的，主要是完成接下来形参到实参的置换，因此实例化中的参数书写也需遵循一定的规则。

3.1）变量名的置换。所谓变量名置换，即只对具体题目参数中的名称进行替换。如，实例化中参数为点 P，题目中具体参数为点 A=(1，2)，变量名置换法则是只替换 A=(1，2) 的变量名”A “，返回的置换结果是 P=(1，2)

3.2）全称替换。有些情况下，需要获取到原始题目的所有参数信息，而不是只进行变量名的替换，需要注意运算符两边变元数相同，全称替换在函数中应用较为广泛。例如，实例化中参数是 $f(x) = g(x)$, 题目中具体参数为 $h(x) = a * x^2 - b$, 因为运算符”=“两边的变元数相同，发生的事全称置换，置换后的结果是保留原始题目参数 $h(x) = a * x^2 - b$；若实例化参数改写成函数 $f(x)$, 则发生的置换是变量

表 3-5 操作函数表

操作函数名	参数表	描述
Simplify	String expression, List<String> conditions	在指定条件下化简表达式
Solve	List<String> equations , List<String> vars	解单个方程或者方程组
rsolve	List<String> condition	解递归方程
union	String setA, String setB	有限集合的并集
intersect	String setA, String setB	有限集合的交集
sumOfPreNminus1Items	String formula	连加数列求和
VectorialAngle	String vector1, String vector2	计算向量夹角
getCosOfVectorAngle	String vector1, String vector2	计算向量夹角余弦值
CalculateMaxValueOfFunc	String funcAndConditions, boolean isMax	在定义域范围内，计算函数最大值
CalculateMinValueOfFunc	String funcAndConditions, boolean isMin	在定义域范围内，计算函数最小值
definitionOfFun	String function, String var	计算函数定义域
extreme2Equation	String function, String extremePoint	根据函数极值点，生成导函数为 0 的方程
diffOfFun	String function， String var	函数求导
PluralPoint	String express	复数求对应点
PluralReal	String express	复数求实部
PluralImag	String express	复数求虚部
equationDiscriminant	String equation	生成方程判别式
slopeEquation	String function	求形如 y=f(x) 的斜率

名置换，置换结果为$f(x) = a * x^2 - b$

3.3) 变元置换。变元置换是指对表达式中的变量进行一一映射，从而产生一组或多组轮换关系。变元置换在置换策略中应用广泛，在完成变量抽取后，对变量进行轮换映射参与后续的计算模块。本文中的变元置换具体表现为：数列角标的置换、函数的自变量置换、系数置换等。例如，$f(x) = a * x^2 + b * x + c$ 和 $f(x) = x^2 + 9 * x - 3$ 所对应的一组轮换映射为 {a=1,b=9,c=-3},$a_m + a_n = p$ 和 $a_3 + a_5 = 0$ 所对应的一组轮换映射为 {m=3,n=5,p=0}

3.3.4 定理的应用

在整个问题求解系统中，解题是最核心的部分，是系统的功能需求和第一步目标，而解题的核心部分是推理引擎，图推理引擎设计的驱动依据就是实例化定理。定理驱动方式分为两种，一种是针对所有实例化定理采用循环驱动，这是一种外部驱动；第二种针对单个实例化，是内部驱动方式。图推理引擎会对实例化定理子图进行解构，将实例化子图拆分成逻辑上独立的两个子图：条件子图和结论子图。其中，条件子图作为引擎匹配的依据，结论子图作为引擎产生新知识以及如何插入到原知识库的依据。当匹配成功时，才会触发引擎对结论子图的处理，否则会进入到外部循环驱动，进入到下一条规则的匹配。因此定理是引擎解题的重要依据，包括如何在定理库中搜寻一条或多条解题路径。

实例化定理除了在解题中应用，同样应用在知识点标注上。在自然语言处理过程中，对实例化定理添加了三个个字段”instantiatedKnowledge“，”instantiatedKnowledgeWeights“,”instantiatedDescription“。”instantiatedKnowledge“代表该实例化定理涉及到的相关知识点，”instantiatedKnowledgeWeights“标记该知识点的权重值，当一个实例化包含多个知识点信息时，可以通过知识点权重来确定知识点的优先级。知识点来源于初高中教材的大纲知识点，由于一条定理可能包含多个知识点，因此字段”instantiatedKnowledge“设计成 List<String> 数据结构。”instantiatedDescription“表示实例化的定理名称，主要是用于后续解答过程的答案输出，同时实例化标签 label 加上定理名称便可形成对定理的唯一标识。解题是知识点标注和类人过程输出的基石和前提，它们之间的关系具体表现为：推理引擎在对实例化定理库进行搜寻的过程中构建解题的定理树，对定理树进行回溯查询得到类人解答过程，定理树是记录解题过程中所使用的实例化定理之间的逻辑关系，通过”instantiatedKnowledge“字段输出本题涉及到的所有知识点内容。

3.4 本章小结

本章首先研究了时域积分方程时间步进算法的阻抗元素精确计算技术，分别采用 DUFFY 变换法与卷积积分精度计算法计算时域阻抗元素，通过算例验证了计算方法的高精度。

第四章 图同构的数学推理引擎的设计和构建

4.1 系统概述

在研究设计图匹配推理引擎之前，先介绍一下《初等数学问题求解关键技术及系统》课题。该系统的研究内容主要包括初等数学问题的题意理解和初等数学问题自动求解两大部分。题意理解就是面向初等数学领域的自然语言处理，会根据数学领域的自然语言特性，研究针对数学领域的中文分词、词性标注、命名实体识别、关系抽取、句法分析等自然语言处理基础技术。另一部分，主要就是根据初等数学问题自然语言理解之后的语义表示，结合问题归约、认知模型、自动推理、大数据深度学习、图嵌入、符号计算等技术，自动求解问题并重构类人解答过程。从上述的介绍中可以看出，图匹配推理引擎主要完成整个系统的第二部分工作，是自动推理的核心部分。本文最终设计了计算推理与逻辑推理是交互式的推理思想，两种推理方式相辅相成，让整个推理引擎的功能更加强大，解题的覆盖面也更加广泛。本章后续的具体小节，将会对图匹配推理引擎与符号计算推理做出详细深入的研究阐述。类人解答系统的结构和流程如图 4-1 所示。

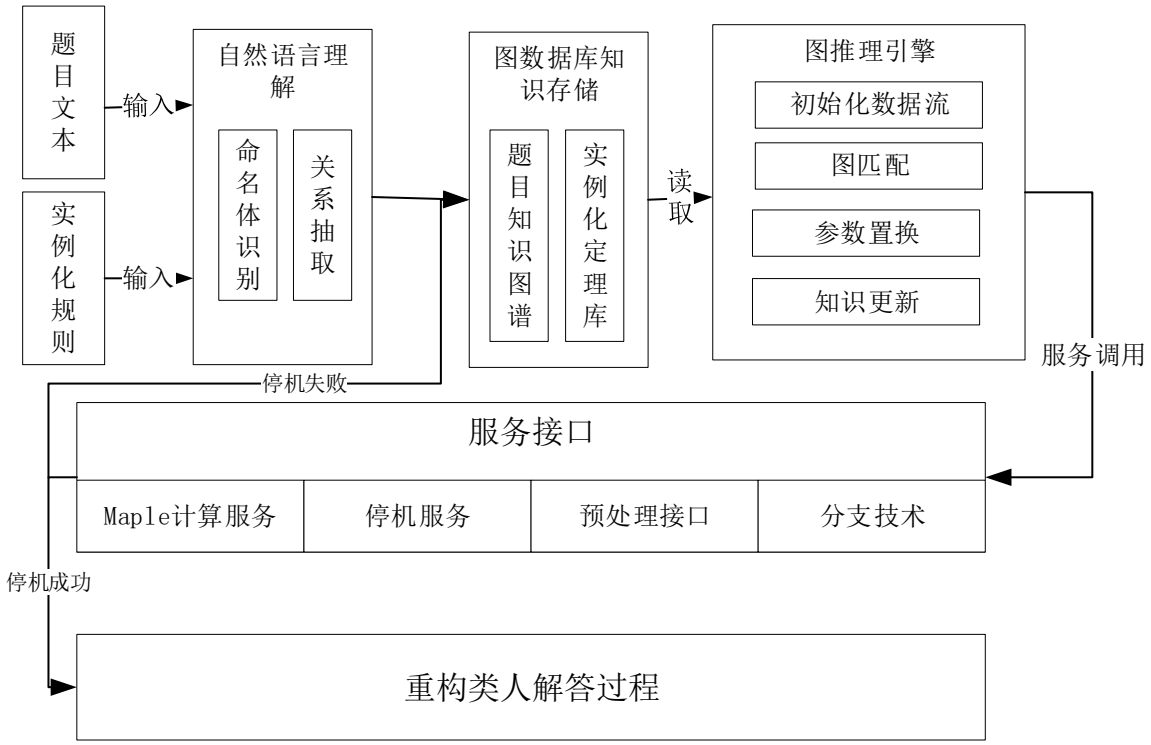


图 4-1 基于初等数学的类人解答系统

从系统架构图中可以看出，类人解答系统共分为 5 个部分：系统的输入和输出、知识理解、知识存储、以及自动推理。系统输入包括两部分，第一部分是待求解的题目文本，这里的文本是需要处理文本中公式和符号，系统接受的文本输入是经过 latex 处理过的文本。同样的，对于实例化规则的文本描述也需要将其中的公式转成 latex 格式。知识理解主要是指利用自然语言理解的相关技术去生成机器可以理解的形式，自然语言理解通过命名体识别和关系抽取的技术手段生成 < 实体-关系-实体 > 三元组的来表示每一句描述的语义。而一段文本则用三元组数组的形式去存储，最终为了前后端交互方便，自然语言的结果以 json 的数据格式进行保存和跨平台传输。后端接受到前端自然语言理解的 json 数据后，进入知识存储和表示模块。为了发挥图的优势，选择使用知识图谱来表征知识，存储在图数据库中。无论是题目 json 文件还是实例化规则的 json 文件都以知识图谱来存储知识，然后存储在 neo4j 图数据库中，不同的是实例化规则是提前批量将 json 文件生成存储在图数据库中，形成实例化规则库，而题目是动态的生成知识图谱并存储，在正确求解完之后将原题目的图谱删除掉。

自动推理和求解主要是依赖于推理引擎去实现。本文设计和构建的推理引擎是基于图同构的匹配置换算法的实现，并融合了计算和逻辑双重交互推理方式。整个推理引擎的设计使用分层结构，共分为四层逻辑结构，每一层相互独立，又层层递进并存在一定的调用和依赖关系。第一层，是读取题目图谱和实例化图谱，对图谱进行解构成内存中最小的推理形式-三元组，因此用三元组集合逻辑上来表示图。在解构的同时会完成内存中各种推理数据结构的初始化工作。第二层，是对题目图谱和实例化规则图谱进行匹配，匹配的结果分为两种：匹配成功和匹配失败。匹配成功会进入到下一层逻辑，匹配失败会停止当前实例化规则的推理，继续去实例化库中选取新的实例化定理。第三层，参数置换层是完成实例化形式参数到题目具体参数的映射。第四层，知识更新层是对第三层生成的新知识进行知识的插入更新。计算推理主要是在推理引擎的第三层逻辑中提供相对应的计算服务，计算服务则是依赖于符号计算平台 Maple 作为基础支撑。

4.2 复杂逻辑推理研究

所谓复杂逻辑推理，顾名思义，即多种推理方式、策略的有机结合而形成的一种复合的推理方式。相对于单一的推理方式，复合的推理方式不仅能更加准确、全面的模拟人类的逻辑推理思维过程，还能极大提高系统的解题能力和稳定性。除此之外，更多的推理方式融合会很大程度上提高了系统的容错率和兼容性。传统的复杂逻辑推力组合包括基于规则的正向推理与基于规约的逆向推理，经典分

析法与反证法融合推理等。

4.2.1 正向推理

正向推理是从已知事实出发，通过规则驱动，不断产生新的知识实体的一种推理方式。基于实例化规则的正向推理基本思想是已知事实在规则的驱动下产生新知识实体，然后新的知识插入到原始事实中去，继续规则驱动，这样不断知识迭代的一个过程。知识迭代的终止条件就是产生的知识与求解目标相匹配，表示到达求解目标，求解成功，终止正向推理。如果新产生的知识与目标知识不匹配，则会将新知识插入到原有的知识库中继续实例化规则匹配，直到所有实例化规则均不能匹配出发产生新的知识为止。此时表示正向推理失败，结束正向推理。正向推理的详细推理过程如下图 4-2 所示：

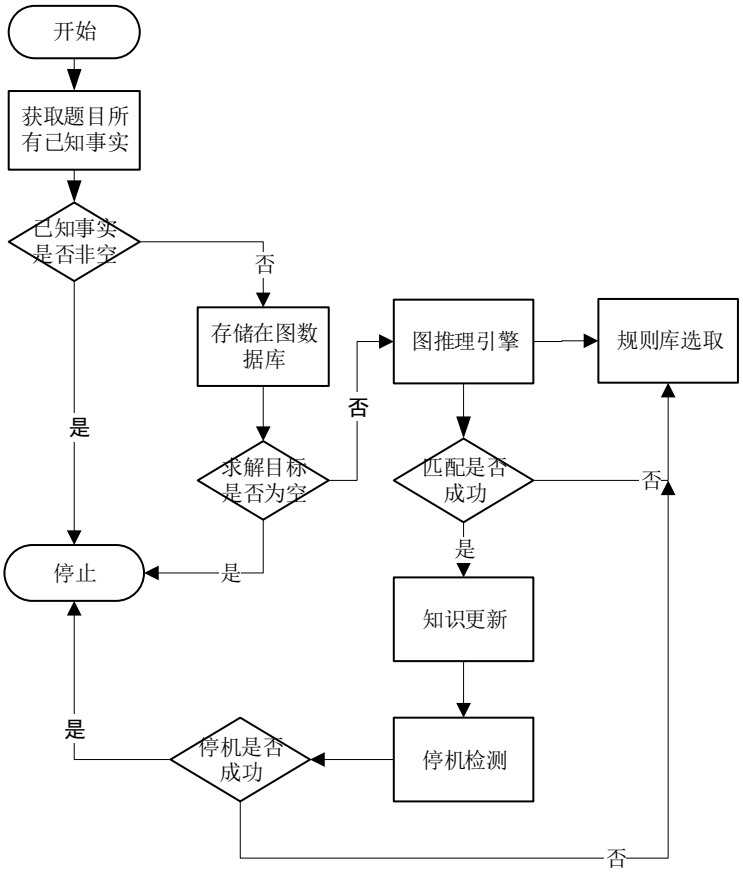


图 4-2 正向推理流程图

由正向推理的流程可以得出，推理的终止条件是在已知事实和求解目标非空

的前提下，停机服务会对每次实例化规则产生的新知识与求解目标比对，若求解目标与新知识一致，这里的一致同样是定义的三元组等价。正确停机是正向推理以及逆向推理的关键，为了保证系统运行的稳定性，不延迟、不错误、不超时是对停机服务基本的要求。

4.2.2 逆向推理

逆向推理是从目标结论出发，采用分解子目标和规约的基本思想，通过对目标结论的等价变换、分解成多个子目标的形式，然后从子目标出发，利用分支的技术，并行规约不同的子目标，一步步递归逆向直到搜寻到已知事实。其算法描述如下：

Data: A set of questionTriplesSet

Result: The tree of rule's node

Begin reverse reasoning;

if *questionTriplesSet == null || all facts is satisfied* **then**
  return;

end

if *questionTriplesSet != null* **then**

  Search conclusion triples in the questionTriplesSet;

    Divide the conclusion triples into multiple sub-conclusion;

end

New Branch reasoning;

if *sub-conclusion triple is contained in the map of triple to rule* **then**

  Get the rule from the map;

    Get the condition triple from the rule;

  if *If questionTriplesSet contain condition triple* **then**

    for *all triple in rule condition triples* **do**

      dfs(triple);

    end

  end

end

算法 4-1 逆向推理算法

4.2.3 正逆结合

正逆结合是一种融合了正向推理和逆向推理的复杂逻辑推理方式，这种混合的推理方式具有兼容性强、推理效率搞、推理方式多样，能较好的模拟人类解题思维过程等优势。与单一的推理模式不同的是，正逆结合的推理方式，可以综合已知事实和目标结论，参与推理的信息更加丰富，同时在推理的过程中两种推理方式也起到互补作用，给各取所长。正逆结合推理分成”先正后逆“、”先逆后正“两种不同方式。本文最终选择了”先逆后正“的方式进行推理。首先，逆向推理是系统的入口，逆向推理的主要功能是提高实例化规则选取的效率，解决正向暴力匹配实例化规则的时间复杂度过高问题。逆推就是通过分解子目标以及等价规约求解目标，然后递归的去搜寻实例化知识库，递归的终止条件是已经搜寻到已知事实，并由此构建一棵推理规则树。然后通过逆向推理构建的规则树回溯出解题路径，将解题路径对应的实例化规则作为正向推理的输入，开始正向匹配，产生新知识。为了进一步提高解题的效率，正推和逆推中都使用了分支并行推理技术。

一般而言，双向推理融合正向推理与逆向推理有三种组合方式：”先正后逆“、”先逆后正“、”正逆同时“。本文采用是”先逆后正“的推理方式，以下小节将详细介绍”先逆后正” 的双向推理流程和算法。

4.2.4 先逆后正

采用“先逆后正”组织结构的双向推理，是将逆向推理作为正向推理的一个重要补充，逆向推理是构建解题路径的规则树，然后再通过回溯算法输出解题的实例化规则路径，路径中的每个节点对应就是正向推理的实例化规则，这个路径作为正向匹配的输入，从而产生每个路径节点对应的新知识。逆向推理相当于根据已知目标结论出发，有目的性的搜索解题中所使用的实例化规则，这样很大程度上提高了正向暴力式匹配所带来的时间开销。

“先逆后正”双向推理算法中，逆向推理最基本的思想是分解子目标和等价归约，然后不断递归这个过程，最终生成包含解题路径的规则树，对于规则树进行回溯生成解题路径，解题路径则作为正向推理的输入。如下图 4-3 是本文的“先逆后正”推理算法思想的框架图。

4.3 图匹配推理引擎的构建与设计

图匹配是个经典的图上匹配问题，图匹配问题涉及二分图匹配、最大连通域匹配、子图匹配等问题。本文设计的图匹配引擎采用的算法思想就是子图匹配算法，又称子图同构算法。所谓子图同构任务，即给定一个 target graph A（m,n）, 和

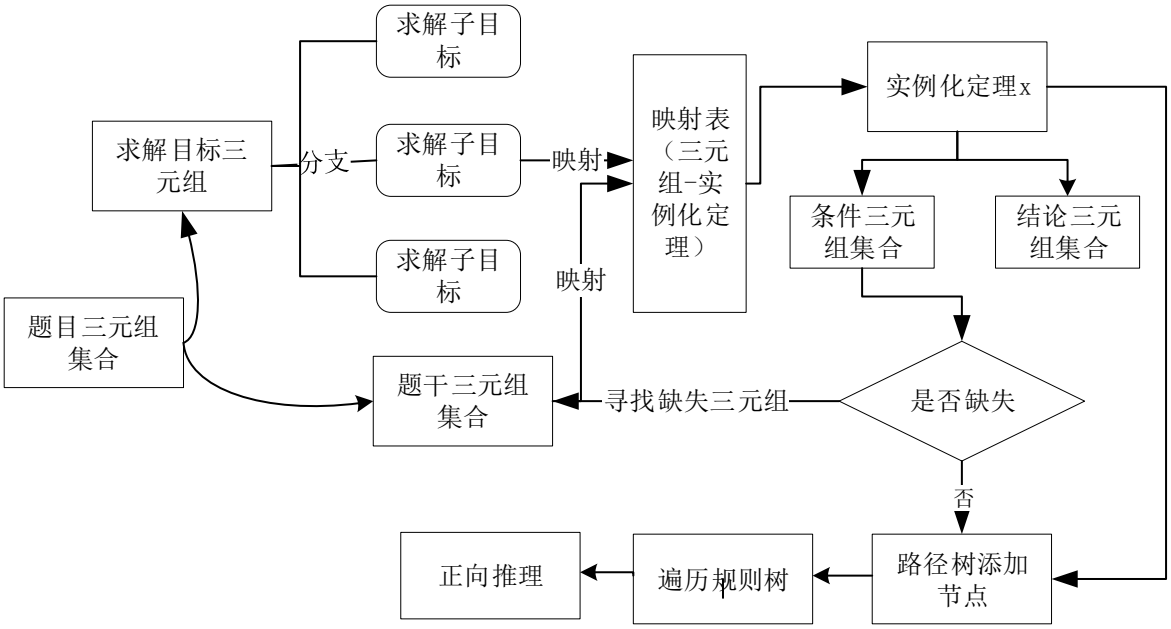


图 4-3 “先逆后正“推理框架图

一个 query graph B(p,q)。其中，m、p 是点集，n、q 是边集。试图寻找一种映射关系，使得两个图对应的点 label 相同，query graph 中的边在 target graph 中对应的点之间都存在，且 label 相同。如下图 3-4 所示：********** 左边图可以同构到右边的上面三角，或者下面三角。

4.3.1 引擎基本设计思想

图匹配推理引擎核心思想是基于图同构的搜索匹配，并寻找一组置换等价的映射集的复杂逻辑推理引擎。上文已经对图同构进行了阐述，图匹配推理引擎由图同构的判断器、置换合一的生成器以及知识冲突消解的处理器三大部分组成。同时也融合了符号计算平台的计算服务、停机等外部接口，为推理引擎提供计算和停机服务。另外，引擎中还引用了分支并行技术，很好的解决了多解、分类讨论、等价变幻、知识分解等一些列较为复杂问题，也提升了引擎并行处理的能力和解题速度。下图 4-5 是推理引擎的详细结构设计图。

从图中可以看出，**********

4.3.2 逻辑架构

图匹配推理引擎共分为四层逻辑结构，每层结构保持相对独立性。第一层逻辑结构，图解构层。该层是整个推理的入口，子图的读取就是引擎的输入，对输

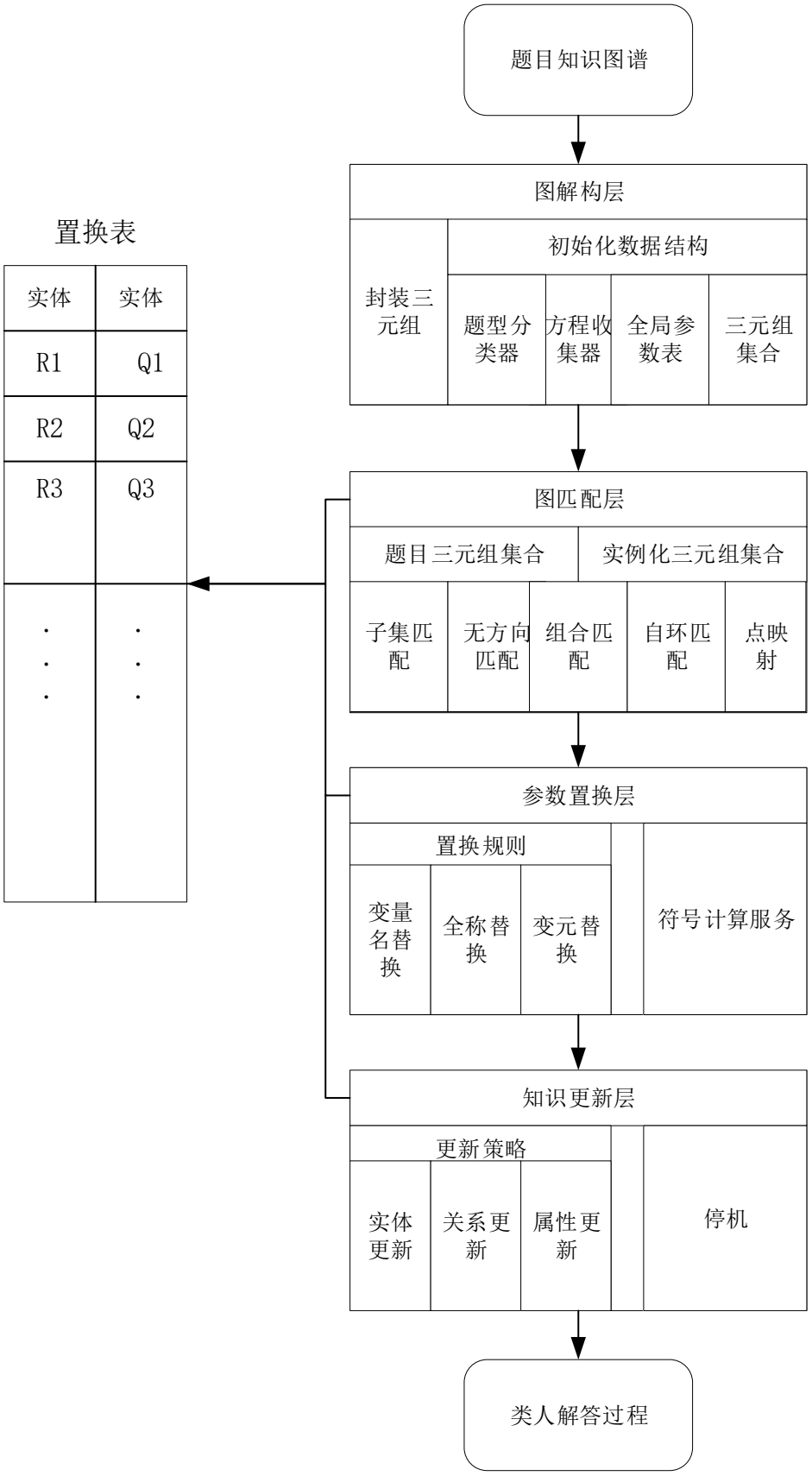


图 4-4 推理引擎逻辑架构图

入的子图进行解构并转换成推理的数据结构，同时初始化内存各种推理数据流。下图 3-6 是推理引擎的第一层逻辑结构图。********** 从图中可以看出，********** 第二层逻辑结构，图匹配层。该层是引擎核心部分，具有图同构的判断器和置换映射集的生成器。第三层逻辑结构，变量置换层。该层依赖于图匹配层，变量置换的是依据置换映射集生成器生成的置换表，找出若干组变量置换。同时，该层还调用符号计算平台服务，置换出的变量映射是符号计算接口的输入参数。第四层逻辑结构，知识更新层。该层是对于变量置换层生成的新知识进行处理，主要包括知识插入和知识冲突消解两种方式。知识插入是完成下一轮匹配搜索迭代的前提，知识冲突消解是包括两个部分，知识异常检测以及知识的冲突检测处理。其中知识异常是指新生成的新知识出现空指针、不合法、矛盾等情况；知识冲突检测是指新生成知识与已有的知识产生冲突，需要相应的策略和规则去处理新旧知识，防止出现知识丢失、冗余、重复。

4.3.3 Match Engine

从上文的阐述中，可以看出匹配是整个引擎设计的核心部分。本节会更加详细的阐述逻辑结构的第二层，图匹配层的整个设计思想和具体结构。下图 4-7 是图匹配详细结构图。

从图中可以看出，**********

4.3.4 匹配原则

本引擎匹配最核心的思想，是寻找两组三元组集合是否满足最小子集思想。通过上文的介绍，题目子图和实例化规则子图都会解构成三元组集合，重新定义三元组 equal 的标准：实体类型相同，关系名相同。引擎将实例化规则三元组集合作为最小子集，在题目三元组集合中搜寻是否覆盖最小子集，为了保证匹配准确度，需要遵循子集匹配、无方向匹配、组合匹配、自环匹配、点映射这五条基本准则。下面将分小节详细介绍每种匹配原则。其伪代码描述为：

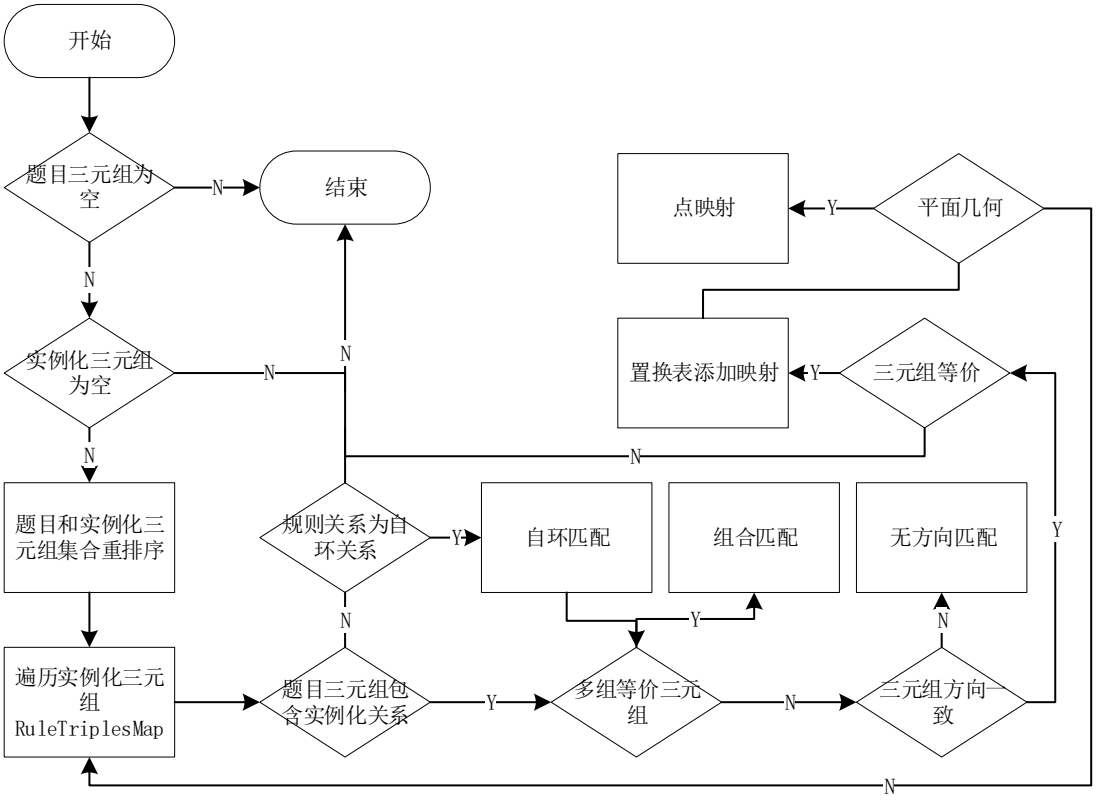


图 4-5 图匹配算法流程图

```
Data: question graph and rule graph
Result: datastream of matching
Begin match reasoning;
   divide question graph and rule graph into
  questionTriplesList and ruleTriplesList;
if questionTriplesList == null || ruleTriplesList == null then
  |  return;
end
Resort questionTriplesList to qMap and Resort ruleTriplesList to rMap;
for all question relation(qr) in rMap's keySet do
  |  for all rule relation(rr) in qMap's keySet do
  |  |  if qMap contains qr then
  |  |  |  Select five matching strateges;
  |  |  |  if rTriple match to multi-qTriples then
  |  |  |  |  Enter into conbinationMatch model;
  |  |  |  |  if Exit the fake match conbination then
  |  |  |  |  |  delete the fake match conbination;
  |  |  |  |  end
  |  |  |  |  Refresh the Map of questionEntity to ruleEntity;
  |  |  |  end
  |  |  end
  |  end
end
if matching is not successfully then
  |  Print some error log about match;
  |  return;
else
  |  enter into next model，continue reasoning;
end
```

算法 4-2 匹配算法

4.3.4.1 子集匹配

子集匹配是保证引擎有较好兼容性和通用性的一个重要原则。子集匹配的对象是三元组中实体类型。子集匹配的定义：给出两组三元组，若它们的实体类型具有继承关系，可以是直接的父类子类继承，或满足间接继承关系，则两组三元组满足子集匹配。下面以一个例子进行说明，题目三元组 < 等差数列-项关系-首项 >，规则三元组 < 数列-项关系-项 >，等差数列直接继承于数列，首项直接继承于项，因此两个三元组满足子集匹配原则，属于可匹配三元组。

4.3.4.2 无方向匹配

无方向匹配是增加引擎匹配搜索灵活性，增大了匹配搜索的范围。知识图谱中存储的题目和实例化规则子图都是以有向图形式存储的，因此三元组均是有方向的。无方向匹配原则是指将三元组匹配进行两次匹配检测，第一次完成正向就检测，匹配成功退出返回 0；匹配失败则将三元组进行逆向匹配检测一次，匹配成功退出返回 0，匹配失败返回-1。例如，题目三元组 < 反比例函数-区间关系-区间 >，实例化三元组 < 区间-区间关系-反比例函数 >，则需要进行两次匹配检测，满足无方向匹配原则，属于可匹配三元组。

4.3.4.3 组合匹配

组合匹配是指一个三元组或一组三元组能匹配上两组或两组以上的三元组集合，这个时候边会出现组合问题。组合匹配形式主要有一对多，多对多的情形。从形式定义和概念上组合匹配较为容易理解，但是如何正确的处理组合分支、组合爆炸是个比较疑难的问题。本文对于组合匹配问题采用了多重解决方案，并在具体题目得以验证和应用。当遇到组合匹配时，基本原则是将组合匹配转换成单一匹配，然后通过循环的方式去完成组合。组合的情况较为复杂，以下将通过例子来列举组合发生的几种情况。

例 1，实例化三元组 < 数列-项关系-项 >, 题目三元组集合 < 等差数列-项关系-首项 >，< 等比数列-项关系-项 >，这是最常见的三元组组合情形，引擎处理的手段是将题目三元组集合拆成等价于最小实例化三元组的多组，然后依次匹配或采用分支技术并行匹配。例 2，实例化三元组并集 AuB-子表达式关系-集合 A，并集 AuB-子表达式关系-集合 B 题目三元组并集 MuN-子表达式关系-集合 M，并集 MuN-子表达式关系-集合 N, 并集 PuQ-子表达式关系-集合 P，并集 PuQ-子表达式关系-集合 Q, 此组合对应于多对多的一种情形，若按照组合学方案会有 6 种组合方式，这样会产生 4 组多余的组合出来，若四组知识继续插入到知识库，进行下一轮的匹配循环迭代会继续产生越来越多的知识，最终便会组合爆炸。如何寻找

有效组合，排除假组合，将在下文详细介绍。

对于上文例 2 的组合情形，引擎会启用三种机制去筛选假组合。方案一：属性标价法。属性标记法，是通过在关系中添加一个属性字段 isGroup 来标记三元组是否属性同一组，同一组的定义是两个或两个以上三元组具有共同的头实体或尾实体。属性标记法通过属性字段来对相同三元组进行分类处理，先分类再匹配，这种方式可以有效的降低组合次数，例 2 中题目三元组四个相同三元组，通过 isGroup 属性标记分类，化分成并集 MuN-子表达式关系-集合 M，并集 MuN-子表达式关系-集合 N 和并集 PuQ-子表达式关系-集合 P，并集 PuQ-子表达式关系-集合 Q 两类，分类完成后将原有的随机组合 6 组降成为 2 组有效组合，筛选掉 4 组多余组合，大大提升了匹配效率。

4.3.4.4 自环匹配

自环匹配是弥补匹配设计缺陷，匹配的最小单位是三元组，因此匹配子图的限制条件确保能解构成三元组集合，不能存在离散的孤立节点。为了让推理引擎通用性更高，引擎对孤立节点情况做了自我完善。当实例化规则子图中出现自环节点时，引擎会自动检测同类型的题目节点并增加一条自环关系，解构后的三元组集合多出相应的自环三元组。

4.3.4.5 点映射

点映射是匹配策略中比较特殊的一种情形，应用场景主要是针对平面几何和解析几何中。在平面几何和解析几何中常常会出现多边形、线段、角、边、点这些描述，且多边形、线段、边、角都包含点的信息，若采用预处理方案将所有的点的信息全部扩展出来，点与边、点与多边形、点与线段之间会有着复杂的关系描述，进行知识表示时会形成较为复杂的图谱，进而会导致匹配时组合爆炸以及后续知识爆炸问题。为了解决平面几何中这一问题，引入点映射简化知识表示。点映射思想是在知识表示环节省去隐含点的信息，简化图匹配的复杂度，生成置换表结构后，再进行点的轮换映射处理，即将图中点的知识扩展问题转化成匹配后的置换表中的点的映射轮换问题。例如图 3-1 所示，图中左边部分是题目图谱，右边是”等腰三角形两腰相等“的实例化图谱。

********** 如表 3-1(a),3-1(b) 分别是点映射前的置换表和点映射后的轮换表。

表 4-1 点映射前置换表

实例化实体	题目实体
A’B’C’	ABC
A’B’	AC
A’C’	BC

表 4-2 点映射后置换表

实例化实体	第一次轮换	第二次轮换	第三次轮换
A’B’C’	ABC	ABC	ABC
A’B’	AC	AC	AC
A’C’	BC	BC	BC
A’	A	B	C
B’	B	C	A
C’	C	A	B

4.3.5 置换等价

置换等价是推理引擎中另一核心问题，处于推理引擎的第三层逻辑结构，它是匹配层和知识更新层的桥梁，起到知识传递作用，同时置换等价的结果作为符号计算服务的参数输入，直接关联到知识产生的正确性。本文结合数学知识图谱的具体特点，在推理引擎中使用的是广义上的置换，为了求解实际复杂的问题，实现形式化参数到实际参数的替换，置换的类型更加多样。

图匹配推理引擎中共使用了五种置换手段：实体置换、分类置换、全称置换、变量名置换、参数置换。实体置换指的是知识图谱上实体和关系的置换，依赖于匹配算法，对于题目图谱和实例化规则图谱匹配生成的实体置换表，这是引擎置换的第一步。分类置换的定义较为简单，是指引擎为了处理不同类型的数据采用不同的置换策略，例如，函数的自变量置换和数列的角标置换是不同类型，需要分类处理，分类置换在引擎中使用广泛。全称置换、变量名置换、参数置换可以统称为字符串匹配中的符号置换，具体三种置换方式的详细解释在第三章的小节中进行了详细介绍和举例说明，三种置换可以看作是字符串符号置换的粒度不一样，其中全称置换的粒度最粗，参数置换的粒度最细。选取哪种字符串的符号置换，同样需要依据具体题目输入的数据和实例化中待置换的数据。如下表是实例化实体二次函数 $f(x)=a*x^2+b*x+c$ 和题目实体二次函数 $g(x)=x^2-3$ 的四种置换策略。

表 4-3 置换等价策略

置换类型	置换前	置换后
实体置换	实体 {name:$f(x)=a*x^2+b*x+c$, type: 二次函数}	实体 {name:$g(x)=x^2-3$, type: 二次函数}
全称置换	$f(x)=a*x^2+b*x+c$	$g(x)=x^2-3$
变量名置换	$f(x)=a*x^2+b*x+c$	$f(x)=x^2-3$
参数置换	$f(x)=a*x^2+b*x+c$	$\{a=1,$ $b=2,$ $c=-3\}$

4.4 知识更新

知识更新是当前轮匹配和下一轮匹配迭代的重要衔接。引擎处理知识更新的技术手段，知识插入、冲突消解、更新回代、自动停机等。知识插入是对实例化规则产生的新知识插入到原有的题目知识图谱中，是知识更新最基础也是最重要的手段。知识更新流程如下图 4-8 所示：

根据图 4-8 所示的知识更新流程图的描述，其对应的知识更新算法伪代码解释如下所示：

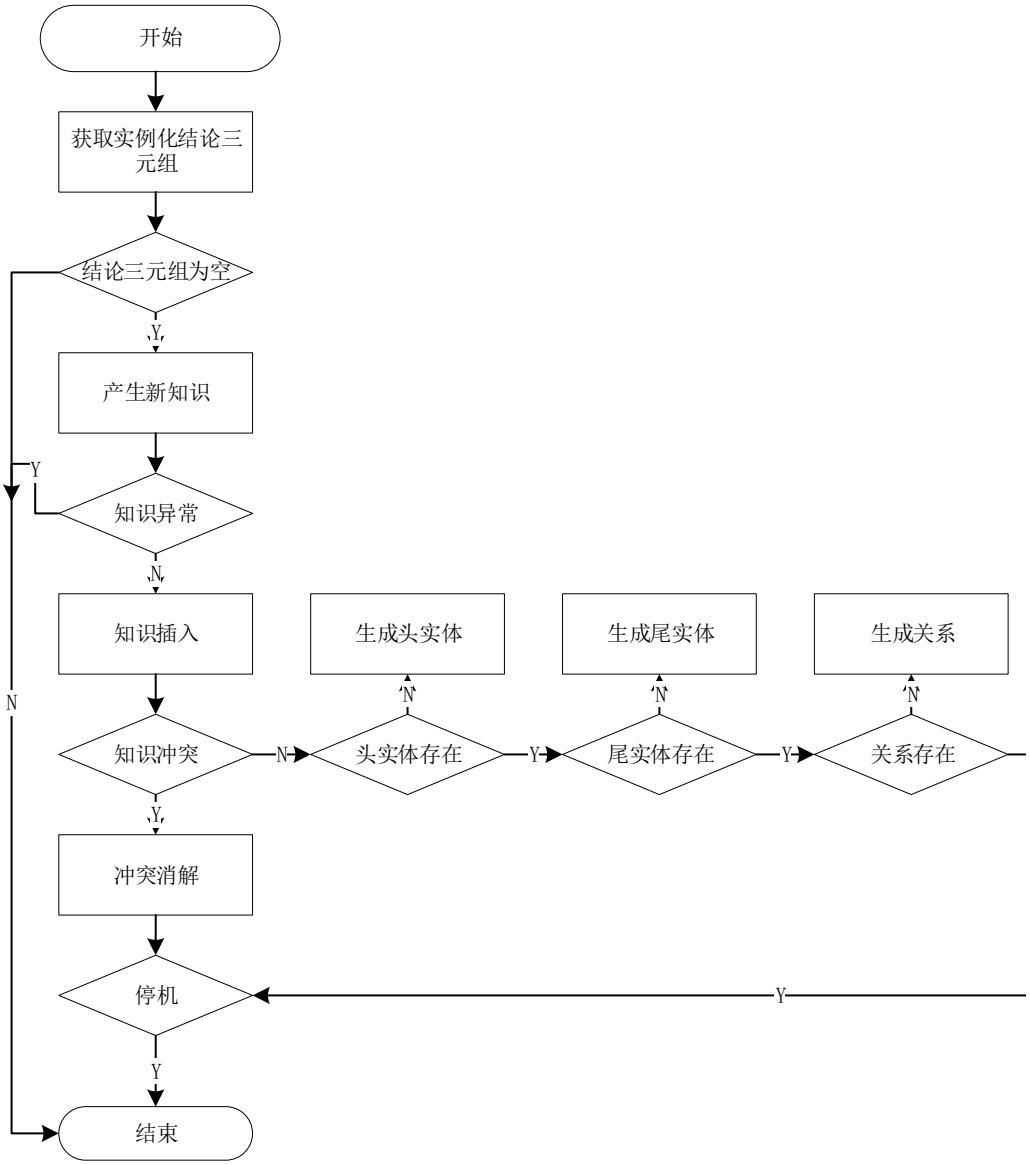

开始
获取实例化结论三元组
结论三元组为空
Y
产生新知识
Y
知识异常
N
知识插入
生成头实体
生成尾实体
生成关系
N
知识冲突
N
头实体存在
Y
尾实体存在
Y
关系存在
N
N
N
Y
冲突消解
停机
Y
Y
结束

图 4-6 知识更新流程图

```
Data: A group of new knowledges
Result: Insert knowledges into question graph correctly
Knowledge exception detach;
if Knowledge is null or illegal then
    end refresh;
    return;
end
if match successfully then
    Get conclusion triple;
      Calculator Integer variety the refreshTag of refresh type;
    if refreshTag > 0 && refreshTag <= 3 then
        select the refresh strateges according to the refreshTag;
        if new knowledge is confilct with exit knowledges then
            solve the confilct betwwen new knowledge and exit knowledges;
            if stop conditions are satisfied then
                end reasoning;
                return;
            end
        end
    end
end
if Knowledge type belong to Equation type then
    Add the Knowledge into Equation Collector;
    return;
end
```

算法 4-3 知识更新算法

从上述算法中，知识更新的依据来源于两部分：第一部分是引擎逻辑层的第三层符号计算平台产生的新知识；第二部分是具体的知识插入策略，在知识更新之前，会将新知识封装成三元组结构，再去解析实例化定理中的结论三元组，结论三元组有 5 种不同情形，每种情形对应一种更新策略，具体更新策略如表 4-1 所示。**********

知识更新是正向推理迭代思想至关重要的环节，算法中冲突消解体现在知识合法性检测上。冲突消解有以下几种情景：

1）新知识为空或者空字符串

2）出现恒等式、矛盾式，如“1=1”，“1=-1”

3）多解，知识冗余

4）新知识插入时已经存在同类型的知识，类型变元相同或者重复都会造成冲突。

4.5 类人解答过程的构建

类人解答过程的构建是类人自动求解系统的输出部分，它是将系统从自然语言理解部分到预处理的变量引入，然后到后端的推理引擎规则推理和计算推理，形成类似人答题的逻辑思维，最终用“因为..., 所以....” 因果逻辑来展示整个推理过程。此模块是依赖于推理模块的逻辑推理和计算推理，在推理的同时构建解答过程规则树，规则树的节点代表使用的规则或者定理，最终的输出就是对规则树进行回溯，得到类人解答过程，适用范围是选择题、填空题、非图表类解答题。输出的形式是两种，第一种是后端以 Log 日志的方式；方式二是知识图谱的形式呈现解答过程子图。

4.5.1 构建规则树

规则树是用树状结构去保存推理过程中产生的路径。树中的节点对应求解中应用的知识，知识来源有原始题目信息生成的节点，推理中使用的实例化定理生成的节点，计算推理产生的方程知识节点，以及表达式的统一化简产生的知识节点。首先，定义树中节点的类结构，**********。具体规则树构建算法流程如下：

1) 将题目中的所有原始节点作为树的第一层节点

2) 进入推理模块，若产生新知识，执行算法步骤 3); 否则继续推理

3) 新知识是否触发分支，若触发分支，开启分支推理; 否则执行算法步骤 4)

4) 将新知识节点添加到树中，并关联其所有父节点，重复执行算法 2-5), 直至停机

4.5.2 重构类人解答

重构类人解答过程一般分为两步：一，在推理过程构建解答过程规则树，并对树中每个节点编号；二，对规则树进行回溯算法，生成可读过程和解答过程子图。节点编号得满足两条约束：1）同一层节点之间的编号按照从左至右的顺序依次编号；2）父节点的编号理应小于子节点编号。编号是在构建节点的同时完成，并作

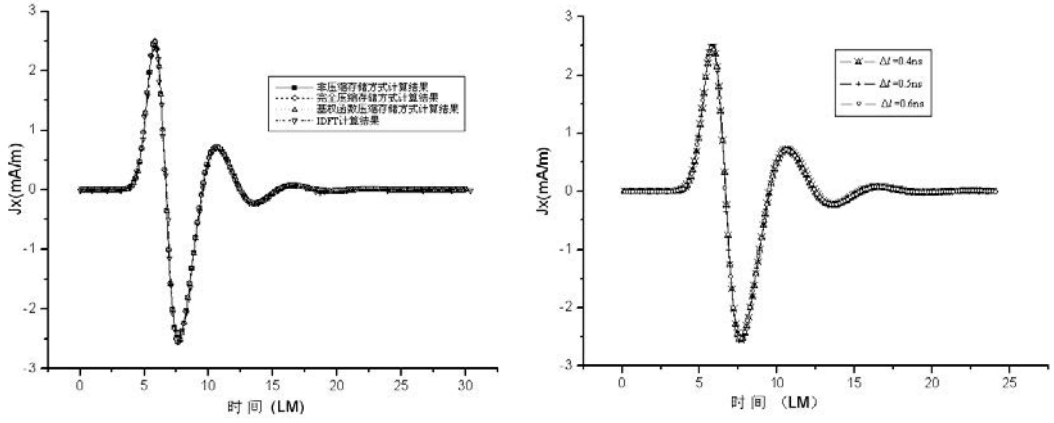
为节点的属性字段存储，方便回溯遍历时输出相对应的编号。类人过程时通过回溯算法去找出树的路径，先深度优先搜索找出对应的因果逻辑关系，然后再回溯找出所有的因果逻辑。若出现一题多解的情形时，利用分支 + 回溯的算法，可以得到多组类人解答过程。

表 4-4 计算 $2m \times 2m$ 理想导体平板时域感应电流采用的三种存储方式的存储量比较。

时间步长	非压缩存储方式	完全压缩存储方式	基权函数压缩存储方式
0.4ns	5.59 MB	6.78 MB	6.78 MB
0.5ns	10.17 MB	5.58 MB	5.58 MB
0.6ns	8.38MB	4.98 MB	4.98 MB

如图4-7(a)所示给出了时间步长选取为 0.5ns 时采用三种不同存储方式计算的平板中心处 x 方向的感应电流值与 IDFT 方法计算结果的比较，……。如图4-7(b)所示给出了存储方式为基权函数压缩存储方式，时间步长分别取 0.4ns、0.5ns、0.6ns 时平板中心处 x 方向的感应电流计算结果，从图中可以看出不同时间步长的计算结果基本相同。

由于时域混合场积分方程是时域电场积分方程与时域磁场积分方程的线性组合，因此时域混合场积分方程时间步进算法的阻抗矩阵特征与时域电场积分方程时间步进算法的阻抗矩阵特征相同。



(a) (b)

图 4-7 $2m \times 2m$ 的理想导体平板中心处感应电流 x 分量随时间的变化关系

由于时域混合场积分方程是时域电场积分方程与时域磁场积分方程的线性组合，因此时域混合场积分方程时间步进算法的阻抗矩阵特征与时域电场积分方程时间步进算法的阻抗矩阵特征相同。

4.6 时域积分方程时间步进算法矩阵方程的求解

定理 4.1 如果时域混合场积分方程是时域电场积分方程与时域磁场积分方程的线性组合。

证明: 由于时域混合场积分方程是时域电场积分方程与时域磁场积分方程的线性组合，因此时域混合场积分方程时间步进算法的阻抗矩阵特征与时域电场积分方程时间步进算法的阻抗矩阵特征相同。 ■

推论 4.2 时域积分方程方法的研究近几年发展迅速，在本文研究工作的基础上，仍有以下方向值得进一步研究。

引理 4.3 因此时域混合场积分方程时间步进算法的阻抗矩阵特征与时域电场积分方程时间步进算法的阻抗矩阵特征相同。

4.7 本章小结

本章首先研究了时域积分方程时间步进算法的阻抗元素精确计算技术，分别采用 DUFFY 变换法与卷积积分精度计算法计算时域阻抗元素，通过算例验证了计算方法的高精度。

第五章 系统测试与分析

5.1 系统测试

本文的测试分为两个部分，第一部分是单例测试，在单例测试中会从最原始的文本输入开始讲解，系统前后端是如何通信的以及前后端协议的数据格式，然后详细介绍系统后端是如何进行问题求解的，各个模块是否完成了对应的功能以及各个逻辑层之间是否完成连通，在此基础上验证系统的整体流程是否流通，保证最后的结果输出和类人解答过程的生成; 第二部分是批量测试，批量测试是随机选定各个不同知识模块 200 题，共 1000 题形成的测试集，其中涉及到函数、数列、向量、解析几何、平面几何等知识模块，系统在测试集上进行测试，通过统计测试的结果，来反馈系统中存在那些不足，对系统的缺陷进行深入分析，归类错误类型，从而增强系统的稳定性、可靠性和兼容性。另外，批量测试还能够去检验概念知识图谱和实例化定理库的完备性。

5.1.1 单例测试

单例测试是从一道题目的原始描述文本开始进行输入，依次进行不同模块测试：题目文本的自然语言理解，这里是测试自然语言理解中的命名实体和关系抽取是否正确；题目 Json 数据生成知识图谱中的子图，这里是测试生成的题目子图知识信息是否完整；预处理模块的知识扩展，测试预处理模块对隐含知识的扩展是否正确；图同构的推理引擎，这里主要是测试推理引擎的三个逻辑层的具体实现模块是否正常运行，三元组匹配模块、变量置换模块、知识更新模块能否连通成功；类人解答过程，测试系统是否正确停机并且输出类人解答过程。

本次单例测试选取的题目是全国卷三文科的第 17 题，具体题目描述为”设等比数列 a_n 满足 $a_1 + a_2 = 4$，$a_3 - a_1 = 8$，(1) 则 a_n 的通项公式为;(2) 记 S_n 为数列 $b_n = log[3](a_n)$ 的前 n 项和，若 $S_m + S_(m + 1) = S_(m + 3)$，求 m “。首先对题目进行分析，第一问是求解等比数列的通项公式，由通项公式的公式 $a_n = a_1 * q^(n - 1)$ 可知只需求出首项 a_1 和公比 q 即可，题干中已经有首项 a_1，因此第一步是通过实例化定理引入公比 q, 然后再将等比数列的项转换成首项和公比表示，得到指包含首项 a_1 和公比 q 的两个方程，求解方程组便可求解出 a_1 和 q，进而求出通项公式; 第二问的难点在于复合数列 $log[3](a_n)$，依据第一问的通项公式可得到复合数列的通项公式，在利用前 n 项和 S_n 与通项公式之间的关系，分别表示 S_m、$S_(m + 1)$、$S_(m + 3)$, 解方程即可得到变量 m 的值。该题的解答过程如图 5-1 所示。 前文详细

开始第一问求解
由题意得：
a_1+a_2=4，a_3-a_1=8 (1)
解方程得：
-8+a_3=4-a_2 (2)
由”扩展公比“实例化定理得：
引入变量q_a (3)
由”求等比数列通项公式“得：
a_n=a_1*q_a^(n-1) (4)
由”通向公式求项“实例化定理得：
a_2=a_1*q_a,a_3=a_1*q_a^2 (5)
联立（2）（5）解方程可得：
a_1=1，q_a=3 (6)
将（6）代入（4）化简可得：
a_n=1/3*3^n (7)
开始第二问求解
由题意b_n={log[3](a_n)}与（7）化简得：
b_n=n-1 (8)
由”通向公式求前项和“实例化定理得：
S_n=1/2*n^2-1/2*n (9)
由题意S_m=-S_(m+1)+S_(m+3)，（9）得：
1/2*(m+3)^2-1/2*(m+3)=m^2 (10)
由（10）解方程得：
m=6 (11)
停机，求解成功!

图 5-1 类人解答过程

介绍过实例化定理库是推理引擎重要的解题依据和驱动器，通过刚刚对题目信息和求解目标的分析，首先确保实例化定理库中包含解题所涉及到的知识定理或公式。本题具体使用到的实例化定理有”数列扩展公比“,”已知首相和公比表示等比数列的项“,”求等比数列的通项公式“,”复合数列的通向公式“,”已知通项公式求前 n 项和 S_n“,”“。对应的实例化描述如表 5-1 所示。

表 5-1 数列实例化定理

定理标签	定理名称	描述
myBkFsvN	数列扩展首项	已知数列 $\{k_n\}$, 则 $\{k_n\}$ 的首项为 $\#firstItem\#\{k_n\}$
ieokuALX	扩展公比	已知数列 $\{k_n\}$, 则 $\{k_n\}$ 的公比为 $\#qName\#\{k_n\}$
RwMkuGfC	已知通向公式求前 n 项和	已知数列 $\{k_n\}$ 的通项公式为 $\{k_n\}=f(n)$, 则 $\{k_n\}$ 的前 n 项和为 $\#sum\#\{k_n\}=f(n)$
jAkkuGfC	求数列的前 K 项的和	等差数列 $\{k_n\}$ 的首项为 k_1, $\{k_n\}$ 的公差为 t, 则 $\#simp\#k_m=k_1+t*(m-1):k_1\&t\&k_m$
DegkuGfC	求等比数列通项公式	$\{k_n\}$ 为等比数列, $\{k_n\}$ 的首项为 k_1, $\{k_n\}$ 的公比为 q, 则 $\{k_n\}$ 的通项公式为 $\#simp\#\{k_n\}=k_1*q^(n-1):\{k_n\}\&k_1\&q$

单例测试是在测试平台上进行的，测试的输入是由题目的输入和实例化定理的输入两个部分组成。其中，题目的输入在测试平台上包含两部分，已知事实和求解目标在网页端分开输入；实例化规则的输入也分成两部分输入，前提条件和结论。接下里，以 3 号实例化规则为例，条件为”“，结论为”“，在网页端的传入如图 5-2 所示，选择的条件和结论的三元组如图 5-3 所示。本题部分实例化定理对应的实例知识图谱分别如图 5-4(a)、5-4(b)、5-4(c)、5-4(d) 所示，

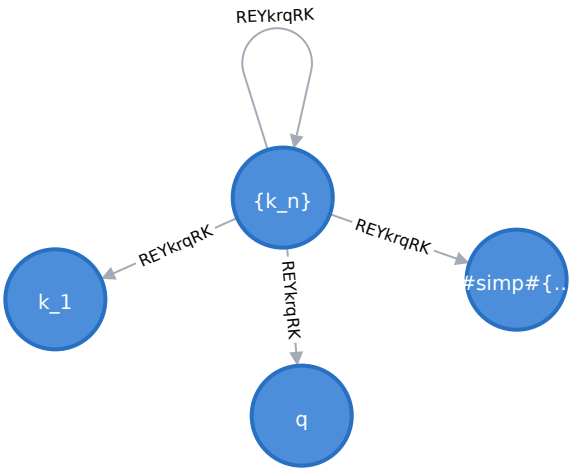


图 5-2 求前 k 项实例化图谱

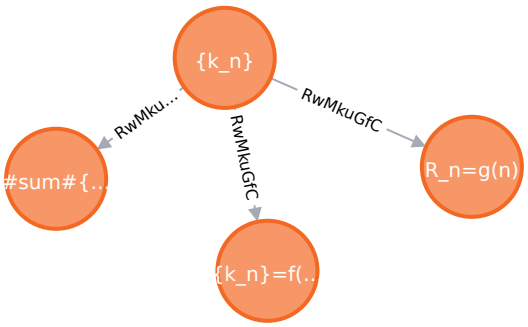


图 5-3 求前 n 项和实例化图谱

在 neo4j 图数据库中添加完实例化定理后，继续对题目的文本信息进行输入，题目的已知事实和求解目标再网页端的输入如图 5-5(a) 所示。题目输入完成后，会在网页首先显示自然语言理解的结果, 是以 json 的数据格式存储三元组信息，具体 json 数据如图 5-5(b) 所示。题目的 json 解析的结果是由三部分组成的，第一部分是自然语言命名体识别的结果，它最终是以一个”entities“数组的形式记录解析结果，构成 json 数据的第一部分；第二部分是关系抽取的结果，即抽取题目中涉及的所有实体之间的关系，最终生成了三元组数组，其中三元组是有所分类的，题干三元组”SteamRelation“以及小问三元组“substeamRelation”, 在形式上可以通过三元组中的“asking”字段进行区分，题干三元组的“asking”均为 0，而小问三元组中的“asking”是”1-8“之间的一个数字；第三部分是一些附加信息，为了前后端通信所添加的字段，”questionId“表示题目的唯一标识，”fakeText“表示题目的文本描述，在后续的类人解答中可以应用。整个题目的知识图谱如图 5-5(c) 所示。 ********** **********

题目实例子图生成之后，将进入后端的推理模块，推理引擎会去 neo4j 图数据库中读取题目子图，读取的方式是根据题目提供的唯一标签, 即”questionId” 这个字段去获取子图。由前文 4.3.1 节中所介绍推理引擎的逻辑架构可知，第一层逻辑层就是解构题目子图和实例化子图，并完成内存中推理的各类数据结构的初始化工作。“QuestionTripls”是保存题目图谱中的所有已知条件三元组，包括求解目标的三元组，设置成全局变量，具体的三元组信息如表 5-3 所示。

推理引擎的第二层逻辑结构是匹配模块，在匹配之前会对初始化的三元组集合进行重排，重排的依据是依据关系中的字段“Order”进行排序，主要目的是为了确保相同三元组有个相对的序，这个序来源于原始的题目描述，通过“Order”字段记录描述的先后顺序。对题目三元组集合和实例化三元组集重排后封装成 map 的数据结构，其中 key 为三元组中的关系，value 值为包含该关系的所有三元组并且有序，因此最终选择的是优先队列进行存储有序三元组。如表 5-4 和 5-5 所

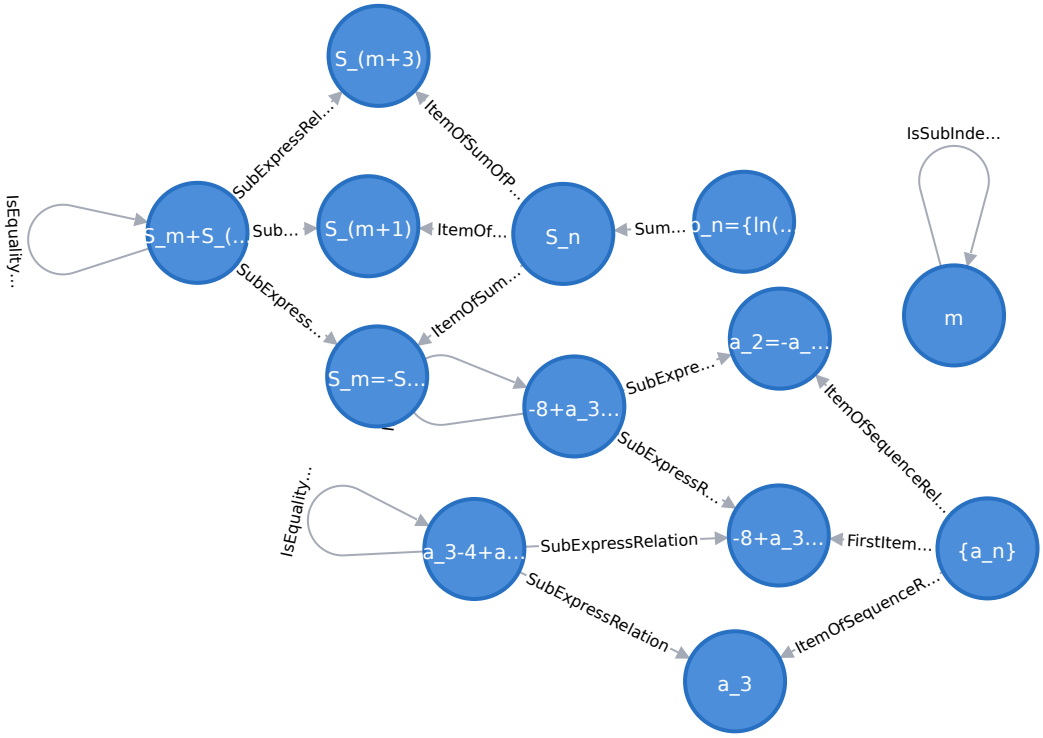


图 5-4 题目知识图谱

示。分别为题目和实例化定理的匹配数据结构。若实例化定理匹配成功，引擎会生成一张实例化定理中形式参数映射至题目中具体参数的置换表，为了保留更多的信息，置换表的 key 和 value 值均为实体类型的数据，每一个匹配成功的实例化定理都有一张置换表，实例化置换表对于后续的推理至关重要。如表 5-6 是“3”实例化的置换表。********** ********** **********

匹配成功后，会进入知识的产生和更新模块。其中知识的产生包括包含两个部分，首先需要进行参数置换，即对实例化中参数转换成具体的题目中参数，置换完参数后需要调用计算平台服务，计算后的结果会进一步封装成三元组，最终新知识同样是以推理最小单位三元组的形式存储。新知识产生的关键一步在于参数置换，而参数置换就是依赖于匹配逻辑中的置换表。因此置换表是整个推理引擎中的重要数据结构，在引擎中各个逻辑层都有所应用。新知识产生后在内存中以三元组的形式存储，同样需要将新知识插入到原始的题目知识图谱中，实例化“3”具体的插入方式如下：首先解析得到实例化的结论三元组部分，如图 5-8 所示，结论三元组的头实体在置换表中可以找到对应的映射实体，属于旧知识无需新生成实体，尾实体是计算服务得到的新知识，在图谱中生成新的实体，实体名为“”，实体类型与实例化的尾实体类型相同。关系“”是原有知识图谱中缺失关系，需要在头实体和新生成的尾实体之间插入。********** 由上述讲解，参

数置换和知识更新均依赖于置换表结构，对于每一个实例化定理都有不同的置换表和更新策略。本题匹配成功的实例化知识更新情况如表 5-7 所示。图匹配完成知识更新和正确停机，求解成功之后的图谱如图 5-6 所示。

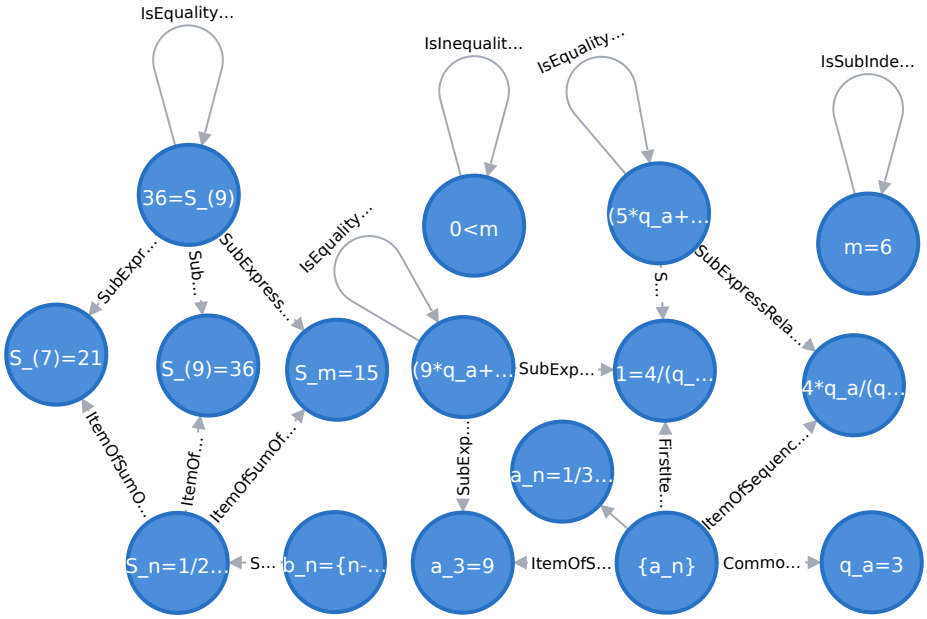


图 5-5 题目更新知识图谱

5.1.2 批量测试

5.2 后续工作展望

第六章 总结与展望

6.1 全文总结

本文主要工作是设计和实现了基于一个图同构的初等数学的推理引擎。工作内容主要分成三个部分：知识图谱构建、实例化定理库构建以及图推理引擎的逻辑结构设计和模块化构建。知识图谱是知识表示的载体，因此构建一个合理完备的初等数学知识图谱是自然语言理解和自动推理的关键一步。知识图谱的构建分为实体构建和关系构建，然后将构建的实体类和关系类通过程序化方式生成对应的图节点和边，存储在 neo4j 图数据库中。为了实现添加实体类和关系类的可扩展性和开闭原则，构建了抽象实体类和抽象关系类，所有的实体类都继承于抽象实体类，所有的关系类都继承于抽象关系类。第二部分是实例化定理库的构建，图推理引擎在推理方式上还是规则引擎的一种，因此规则库是引擎驱动和产生新知识的重要依据。实例化定理库的构建分为三个环节：定理收集、定理标准化以及定理实例化。定理收集是原始教材或教辅资料中的定理、公式、公理等描述，还有一部分是来源于标准答案中的解题过程；完成定理生成后，需要对定理进行规范化、标准化书写，这里的规范化是指符合推理引擎的定理输入规范格式，方便推理引擎统一处理；定理实例化是将第二环节中的规范化定理生成知识图谱，存储在规则库中。最后是图推理引擎的整体设计和实现，引擎在架构设计思想上采用的分层结构，不同逻辑层之间相对独立，模块化程度高；在算法上的核心设计思想是图匹配以及置换等价问题。最后为了提高引擎的处理问题的能力和稳定性，引入异常检测机制和冲突消解策略。综上可以将本文中的研究工作及成果分为以下四个要点：

（1）构建了知识图谱中实体类和关系类，并用抽象类的设计思想实现了知识图谱构建的可扩展性，并定义了抽象类的属性和结构，最终构建的知识图谱实体********** 个，关系 ******** 条

（2）参与了实例化定理的搜集工作，制定了定理的规范化标准，并完成定理实例化，共有实例化定理 ********** 条

（3）完成了整个推理引擎的设计和构建工作，包括其中的逻辑架构设计和实现，核心算法的设计和实现，使用的推理方式设计和实现，每层的模块化构建，异常检测机制和冲突消解策略

（4）参与了类人解答过程的算法核心思想设计和部分构建工作。设计类人解答过程的分支回溯思想，以及推理中的规则树构建。

6.2 后续工作展望

虽然本文完整了整个图推理引擎的设计和构建工作，并且也将推理引擎接入到系统中完成测试工作。但是引擎在稳定性和兼容性上还存在不足，解题能力也有待提高。由于时间和精力有限，推理引擎中存在的不足和疑难问题还待解决：

（1）匹配算法的精度问题。目前引擎中使用的匹配粒度是三元组，即三元组是匹配的最小单位。而三元组采用是类型匹配，当将知识图谱拆成三元组的粒度时，会使图的部分信息丢失。后续工作中准备采取关联三元组来解决这一问题，让三元组与三元组之间也产生关系依赖。

（2）知识爆炸问题。因为引擎是产生式系统的演变，因此会存在知识迭代，若匹配过程中出现 m 个事实对应 n 个实例化规则模式，将出现组合情况，组合的知识继续参与迭代，迭代次数过多将出现知识爆炸。后续工作准备在检测机制中加入虚假组合的检测，将无用组合通过检测机制消除掉，减少知识迭代。

（3）知识冲突问题。推理引擎的第四逻辑层是知识更新，知识更新是完成将新知识插入到原始的知识库中。当知识库存在于待插入的知识同类型的知识时，会出现知识冲突，如果不解决知识冲突问题，将会导致知识插入失败，推理迭代出错，甚至会出现推理异常终止。后续工作时继续完善知识冲突消解策略，能够处理和兼容到更多的知识冲突问题。

（4）推理效率问题。在引擎具备一定的稳定性和兼容性的前提下，需要考虑推理效率问题。推理效率一般以时间效率和空间效率作为衡量标准。目前系统中影响推理效率主要来源于三个方面：实例化定理的选取问题、分支组合数、引擎本身的算法。引擎中主要推理方式还是正向推理，而正向推理是暴力搜索的方式，会出现很多无效匹配；分支组合数过多时，知识迭代会引起知识爆炸，影响推理效率；引擎的算法需要优化核心的匹配算法。

致 谢

在攻读博士学位期间，首先衷心感谢我的导师 XXX 教授

参考文献

[1] 李佳珂. 人工智能对金融创新的积极影响 [J]. 人民论坛, 2018, 606(25): 80-81

[2] 吴文俊. 初等几何判定问题与机械化证明 [J]. 中国科学, 1977, 20(6): 507-516

[3] 宋慧欣. 奇点临近, 人工智能大爆炸 [J]. 自动化博览, 2016,

[4] 严佟然, 李晓霞. Grobner 基方法与吴方法在平面几何定理机器证明中的应用与比较 [J]. 海南大学学报 (自然科学版), 2004, 22(002): 111-115

[5] 廖文访, 许明春. 关于 steiner-lehmes 定理的 4 个推广猜想的研究 [J]. 南京信息工程大学学报, 2009,

[6] 靖争. Xml/schema 到 owl dl 本体映射的研究 [D]. , ,

[7] D. Amerland. 谷歌语义搜索 [M]. 谷歌语义搜索, 2015

[8] 汪磊. 基于贝叶斯网络算法的适应性网络教学平台规划 [J]. 商情, 2011, 000(047): 166-167

[9] 张平, 蓝海林, 黄文彦. 技术整合中知识库的构建研究 [J]. 科学学与科学技术管理, 2004, 25(001): 31-34

[10] 张润. 基于 rete 算法的规则引擎 drools 的研究 [D]. , 2016,

[11] 马秀丽, 王红霞, 张凌云. Drools 在网络故障管理系统中的应用 [J]. 计算机工程与设计, 2009, 30(008): 1859-1862

[12] 徐小博. Scada 系统中数据采集与处理模块的设计与实现 [D]. , 2014,

[13] 顾志峰, 董文生, 吴凤品, et al. 业务规则管理系统在金融信息系统中的应用 [J]. 信息技术与标准化, 2013, 000(003): 56-59

[14] 李雄波. 知识表示在教学领域的应用研究 [D]. , 2007,

[15] 韩义. 面向对象专家系统在牛病诊断中的应用研究 [D]. , ,

[16] 孙志森, 李宏欣, 席耀一, et al. 人工智能与神经网络发展研究 [J]. 计算机科学与应用, 2018, 008(002): P.154-165

[17] 汪庆华. 人工智能的法律规制路径: 一个框架性讨论 [J]. 现代法学, 2019, 41(02): 55-64

[18] □. 王玉平. 数据开放式 web 技术探索 [J]. 中国教育网络, , 2 期): 63-66

[19] 刘植惠. 本体 (ontology) 与语义网 (semantic web)[J]. 重庆图情研究, 2006, 007(003): 1-4

[20] 郝惠娟. 基于信息模型的术语库构建与维护方法研究 [D]. , 2018,

[21] 黄恒琪, 于娟, 廖晓, et al. 知识图谱研究综述 [J]. 计算机系统应用, 2019, v.28(06): 3-14

[22] 周君豪, 李源枫, 金明晖, et al. 基于知识图谱的交互式智能儿童玩具设计研究 [J]. 美与时代（上旬刊）, 2019, 000(010): 98-100

[23] 张莹莹. 基于知识图谱的舌像诊疗系统研究与构建 [D]. , ,

[24] 邹银凤. 知识图谱构建中的多数据源实体匹配研究 [D]. , ,

[25] 张杰. 文献结构化的细粒度检索技术研究 [D]. , ,

[26] 马灿. 面向" 智慧法院" 的知识图谱构建方法与研究 [D]. , ,

[27] 沈超. 基于 mpso-kmeans 算法的微博推荐系统研究与实现 [D]. , 2019,

[28] 于方, 刘延申. 大数据画像——实现高等教育" 依数治理" 的有效路径 [J]. 江苏高教, 2019, 000(003): 50-57

[29] 李友江. 基于数据关联的决策数据浏览模式研究与开发 [D]. , ,

[30] 陈婵. 基于关联数据的数字图书馆 web 服务语义关联与发布研究 [D]. , 2018,

[31] 张兆锋. 基于知识图谱的技术功效图自动构建及其应用研究 [D]. , 2019,

[32] 屈峰林. 基于知识图谱的健康医疗知识推送系统研究 [D]. , 2018,

[33] 吕丹阳. 基于关联图谱的高校图书馆图书个性化推荐方法研究 [D]. , ,

[34] 陈品德. 基于 web 的适应性学习支持系统研究 [J]. 华南师范大学, 2003,

[35] 孙晔, 吴飞扬. 人工智能的研究现状及发展趋势 [D]. 价值工程, 2013, 5-7

[36] X. F. Liu, B. Z. Wang, W. Shao, et al. A marching-on-in-order scheme for exact attenuation constant extraction of lossy transmission lines[C]. China-Japan Joint Microwave Conference Proceedings, Chengdu, 2006, 527-529

[37] 王浩刚, 聂在平. 三维矢量散射积分方程中奇异性分析 [J]. 电子学报, 1999, 27(12): 68-71

[38] 竺可桢. 物理学 [M]. 北京: 科学出版社, 1973, 56-60

[39] 陈念永. 毫米波细胞生物效应及抗肿瘤研究 [D]. 成都: 电子科技大学, 2001, 50-60

[40] 顾春. 牢牢把握稳中求进的总基调 [N]. 人民日报, 2012 年 3 月 31 日

[41] 冯西桥. 核反应堆压力容器的 LBB 分析 [R]. 北京: 清华大学核能技术设计研究院, 1997 年 6 月 25 日

[42] 肖珍新. 一种新型排渣阀调节降温装置 [P]. 中国, 实用新型专利, ZL201120085830.0, 2012 年 4 月 25 日

[43] 中华人民共和国国家技术监督局. GB3100-3102. 中华人民共和国国家标准–量与单位 [S]. 北京: 中国标准出版社, 1994 年 11 月 1 日

[44] M. Clerc. Discrete particle swarm optimization: a fuzzy combinatorial box[EB/OL]. http://clere.maurice.free.fr/pso/Fuzzy_Discrere_PSO/Fuzzy_DPSO.htm, July 16, 2010

附　录

外文资料原文

6.1 A Tight Upper Bound on Bit Error Rate

外文资料译文

6.1 基于多载波索引键控的正交频分多路复用系统模型