珠峰学报

Journal of Zhufeng

2024 Spring

钱院学辅 发行

钱学森书院学业辅导中心发行

Published by *Qian Xue Sen College Academic Counselling Center*

珠峰学报

Journal of Zhufeng

主管 西安交通大学钱学森书院
主办 西安交通大学钱学森书院学业辅导中心（钱院学辅）
期号 2024 春（总第 10 期）
封面 腾飞塔（马嘉鸿 摄）
发行时间 2024 年 4 月 8 日
征稿邮箱 qianyuanxuefu@163.com
网站 https://qyxf.site/zhufeng/

发行说明

本刊物同时以纸质版与电子版两种方式发行，板式与内容相同。欲获取电子版，请访问钱院学辅信息站「珠峰学报」栏目（https://qyxf.site/zhufeng/）。

关注我们

钱院学辅微信公众号

目录

校庆特辑 *Special*

珠峰学报（钱院学辅） 1

扬帆 *Sailing*

《珠峰学报》征稿启事（《珠峰学报》编辑部） 4

《珠峰学报》招募启事（《珠峰学报》编辑部） 6

文章速览（《珠峰学报》编辑部） 8

砥砺-自然 *Diving-Nature*

基于热电比拟的二位热传导虚拟仿真实验的开发（郝瑞霆，应雨倩，刘明阳） 9

Fast And Comprehensive Dependency Scanner Using Clang（*Junyang Zhang*） 15

Efficient Embeddings of Logical Variables for Query Answering over Incomplete Knowledge Graphs（*Dingmin Wang, Yeyuan Chen, Bernardo Cuenca Grau*） 20

砥砺-社科 *Diving-Social*

从言与意的张力谈对《庄子》的阅读体认
——以庄惠“濠梁之辩”寓言为例（焦成城） 31

校庆诗词

珠峰学报

钱院学辅

江城子·交大校庆

曲圣

败军甲午震全京。
沪滨兴，上庠[1]生。
扶世济民，无士为功名。
千百师生随诏令，瞻兴庆，定征程。

秦山渭水亦含情。
借东风[2]，绽红樱[3]。
建业百年，日日有佳声。
欲问华章何处续，新港建，更西行。

[1]上庠：古代学堂
[2]东风：表面指春风，深层指新中国改革建设的风。
[3]红樱：表面指交大樱花，深层指交大创造的成果。

贺新郎・在交大的这几年

李君一

记当时、孤蓬北上，浑如风絮。
不解云台如何向，又恨此身孰遇。
此般我、人间无数。
君视流萤犹作月，半顷光、敢拒千年暮。
明与否，何须顾。
往来秋过先生语：
想平生、在乾坤里，与俗名恶。
须记山高天无限，月挂中空谁取。
正当下、荧荧如炬。
别后还知深深意，念君时、身向荒芜去。
擎石者，将天补。

江城子 • 颂西迁

李君一

烟横古戍锁秦川。
镐京孱。望西迁。
夫子先生、亲赴解棋残。
定式腾挪封险象，经甲子，复收官。

沙坡从此贯长安。
柳争妍。引蝶穿。
金色梧桐、时下正当年。
应是落红惜我辈，须饮水，且思源。

热土由东至西往
——兼为西安交通大学校庆祝福

朱凯阳

梧桐落后，樱花开了。
绸缎般的柔软和琳琅的美，
流转在咸宁西路的某处。
如今四海的朋友在四季里相聚，
弦歌自钱学森的年代飘来，
溢满九个书院。瘦小的红船
总摇荡着它巍峨的故事，在一片
土地，青年人们学了去。曾经
热土由东奔赴西，在中华西北
——三秦的土地上，扎根。
渭河水倒映出整个世界的明月光，
从此，长安有了新一种传承。
有时我们称之为西迁，有时
我们称之为自己。
而你知道，
午夜也兜不住流萤，
秋水也盈不满我们将至的远方。

庆祝西安交通大学校庆

支光远

发扬文理仙交处，
励志精读机电书。
立校衷情心不改，
强国血性色如初。
研学济世千家幸，
断垄开封万户福。
此地钟灵英荟萃，
人杰磨戟报千夫。

作者信息

曲圣　西安交通大学智电钱 2101 班学生，东亭诗社成员。
李君一　西安交通大学统计 2101 班学生，东亭诗社成员。
朱凯阳　西安交通大学贸易经济专业 22 级学生，东亭诗社副社长，兼新诗部部长。
支光远　西安交通大学能动 2318 班学生，东亭诗社成员。

扬帆

《珠峰学报》征稿启事

《珠峰学报》编辑部

关于《珠峰学报》

《珠峰学报》是2019年由钱院学辅成员和钱院各班级同学积极筹办的，从整个钱学森学院乃至全校同学中征稿。2021年春，《珠峰学报》的征稿范围扩大至全国重点高校的本科生，以促进各个学校之间的学术与生活交流。《珠峰学报》之名来自于钱院各基础学科试验班所参与的教育部人才培养计划——「珠峰计划」，杂志的定位是：**完全由学生主导、面向国内重点高校同学的综合创新学术刊物，以促进各兄弟院校的学术与生活交流**。期刊将由全校最顶尖的学生排版团队进行设计与排版，同时发布电子版与打印版：电子版PDF文件直接发布于钱院学辅信息站的专栏页面（https://qyxf.site/zhufeng），打印版则在校内定点发放。

板块说明

我们主要按以下三大板块发布作品，文章可以是中文或英文，且允许有多位作者。

- **扬帆** 分享同学们在校园竞赛、社团方面的经验与体会；
- **砥砺** 发布同学们的科研、学术创新成果，是《珠峰学报》的**主要版块**；分为「砥砺-自然」和「砥砺-社科」两个子版块；
- **俯仰** 展示同学们在学业之外的日常思考、感受。

此外，在每一期中可能会安排不同主题的专栏，以响应重大社会事件与活动。以下通过若干例子展示各个版块接收的文章类型：

砥砺 *Diving*

- 科研训练、课外探究等的学术作品、学术写作等；
- **具有一定深度**的课程小论文与大作业；
- 关于自己所在学科前沿问题的报道、综述、思考；
- 简单应用课程知识思考、解决生活中的问题。

扬帆 *Sailing*

- 学科竞赛、科技竞赛的参赛经历、经验分享；
- 出国交流的经历介绍、经验分享；
- 社会实践、课外文艺活动等的成果分享；
- 科研训练中信息检索、仿真工具、实验技巧等的经验分享；
- 托福、雅思、GRE等标化考试的备考经验。

俯仰 *Musing*

- 文学创作、日常思考等内容；
- 对课程安排、课程教学方式、课余活动等的思考、意见；
- 对身边优秀同学、老师的观察、记录与采访。

投稿流程与要求

《珠峰学报》的主要征稿对象为**全国各重点高校的本科生**。在确认个人信息后，我们也接受全国各重点高校的在校研究生、在校任教老师等的投稿。

除纯粹的文学作品之外，要求所有稿件的作者使用**真实署名**，并在提交稿件时附上自己的**学校和班级信息**。登载的文章默认采用CC BY-NC 4.0协议授权，以保护著作权利；如有特殊要求，请在投稿时说明。

稿件审理流程

1. **投稿** 《珠峰学报》全年征稿。请通过审稿总部邮箱 qianyuanxuefu@163.com 投递邮件，将邮

件主题设定为「《珠峰学报》投稿：意向板块 + 文章题目」（如「《珠峰学报》投稿：砥砺 + 有关 xx 现象的研究」），在邮件正文中列出所有作者，及您的**常用的联系方式**，如手机号码或 QQ 等，并将作品文件作为附件发送。初步投递稿件时，可发送 Word 文档、PDF 文档等**易于批注**的文件格式。

2. **审稿及修改** 稿件提交后，经过编辑和审稿人员的审核，将格式或内容修改建议反馈给您；若您的稿件审理意见为「需要修改」，则请您结合审稿人的意见处理稿件，再次通过邮箱回复给审稿人员。同时，为尽快完成审稿流程，如无特殊情况，请您在**一周之内**给出修改稿件的回复。
3. **接收** 若经多次修改后，稿件审理意见为「完全接收」，请您将最终的稿件以 Word、LaTeX 等**可以编辑**的格式提交到审稿总部，编辑将进行排版、校对。
4. **发布** 在当期工作完成之后，我们将通过线上、线下平台发布《珠峰学报》，并通过邮箱联系您，进行奖励发放。

在投稿周期中，请经常检查您的邮箱，与我们保持联系！

学术诚信守则

本刊物要求：所有学术类稿件都应遵守基本的**学术诚信原则**。具体而言，应做到以下几点：

1. 在使用他人的成果，以及自己的已发表成果时，须**详细注明出处**，使用引用标记（引号、文体变化等）明确表明对他人内容的引用。若在本次投稿时使用之前已作为课程论文、大作业等已提交过的作品，需说明之前的**提交情况**。
2. 参考文献原则上使用 **GB/T 7714-2015 标准**填写（英文作品可以 APA、MLA 等国际通行引用格式，但注意一篇文章中的**参考文献格式需统一**）；建议采用 CNKI 等文献工具直接导出对应记录（而非自行撰写）。如文中仅引用了一般性质的网络资源（如博客、日志、新闻等），可以只用较为简明的方式标出引用资料，但需给出真实有效的 URL 及访问时间。

另外，所有投稿作品均需符合社会主义核心价值观，积极正向的要求。

不符合以上基本要求的作品，《珠峰学报》编辑部将不予处理。

投稿奖励

作者的投稿在推动学术规范写作、科研知识推广与实践经验分享等方面具有重大作用与贡献，我们承诺：将根据排版完成后正文的**最终版面数**，对每一份发布于《珠峰学报》上的作品发放电子版稿件**收录证明与稿件补助**，补助标准为：

- 视文章内容与质量，中文每 2000 字 20 元到 30 元，英文每 700 词 20 元到 30 元（上限每篇 150 元）。
- 作为礼品，我们将向每位入选的作者赠送一本当期的纸质版《珠峰学报》。

尾声

一学期很短，而大学生活的四年与之相比甚至还要更短；在有限的时间之中，我们与你一样，追逐着平凡而伟大的梦想。你的梦想，由你自己抒写；我们的梦想，则需要与你一样的众多同学一同参与，一同为之付出辛劳，并在未来一同收获丰硕的果实。

长风破浪会有时，直挂云帆济沧海。

感谢你对于《珠峰学报》与钱院学辅的支持，并祝愿你的远大前程与《珠峰学报》一同起航，驶向无垠的天际！

注：本启事将实时发布于

`https://qyxf.site/zhufeng/call-for-papers`

你可以访问、收藏这一页面。

《珠峰学报》招募启事

《珠峰学报》编辑部

关于《珠峰学报》

《珠峰学报》的前身是由西安交通大学数学试验班自主管理委员会创办的《珠峰报》（创刊于 2017 年）与《东经 109》（创刊于 2018 年）。2019 年，两份刊物的主创人员即将毕业，钱院学辅成员和钱院各班级同学积极组建编委会（下属审稿部与编辑部），筹办《珠峰学报》，从整个钱学森学院乃至全校同学中征稿。2021 年春，《珠峰学报》将征稿范围扩大至全国重点高校的本科生，以促进各个学校之间的学术与生活交流。《珠峰学报》之名来自于钱院各基础学科试验班所参与的教育部人才培养计划——「珠峰计划」，杂志的定位是：**完全由学生主导、面向国内重点高校同学的综合创新学术刊物，以促进各兄弟院校的学术与生活交流**。期刊将由全校最顶尖的学生排版团队进行设计与排版，同时发布电子版与打印版：电子版 PDF 文件直接发布于钱院学辅信息站的专栏页面（https://qyxf.site/zhufeng），打印版则在校内定点发放。

部门介绍

《珠峰学报》主要分为两个部门：编辑部和审稿人团，由一位主编和两位副主编管理。《珠峰学报》仿照国内外期刊的运行惯例，规定各个部门的责任及工作。

编辑部

编辑部的主要工作是监督稿件质量和排版，是《珠峰学报》工作的重点。作者来稿后，主编将文章转发编辑，每位编辑都需要将自己所负责的稿件进行以下流程：

初步检查来稿的格式及内容是否符合规范

编辑应初步检查格式及内容是否符合规范。若格式有问题，如参考文献引用不当、图片不清晰、语句不通顺等，需向作者提出初步格式修改意见；若内容有问题，如违背社会主义核心价值观、体裁不符等，需与主编协商后拒稿。

联系并督促审稿人检查内容

待投稿文章的格式和内容初步符合规范后，联系一位与该文章相关的审稿人，询问是否有时间审稿。如果有，则将待审稿件转交审稿人，并监督审稿进度，直至审稿流程结束。

文章排版

若文章的审稿结论为「完全接收」，编辑需按照《珠峰学报》排版规范，使用 LaTeX 对文章进行排版。

审稿人团

审稿人团的主要工作是负责稿件的内容质量，是《珠峰学报》工作的核心。每位审稿人都应具备至少一个领域的学术视野，在同意编辑的审稿邀请后，均需按照《珠峰学报》审稿规范，对文章进行审核，若不合格需联系作者反复修改，直至「完全接收」。

主编和副主编

主编和副主编的主要工作是，制定《珠峰学报》的运营规范，整理每一期《珠峰学报》的文章，同时确保《珠峰学报》运营流程的实施。

部门奖励机制

每一期发表后，将统一对《珠峰学报》的各个成员发放工作奖励。

编辑

编辑算作钱院学辅干事，出刊后发放工时，根据出刊后负责文章数量统计，每篇文章 6-10 个工时，统计好后提交给学辅负责人。

审稿人

以一位审稿人负责的一篇文章为一个单位，视工作质量，评出甲、乙、丙三个等级，分别对应发放 60、40、20 元，审稿但未接收折扣为 90%。**西安交通大学本科在校生可申请将工资换算成工时发放**。

招募要求

编辑部招募要求

欢迎西安交通大学所有在校本科生有志于与《珠峰学报》一起成长、提升自身学术素养、文字修养、论文排版能力的同学加入编辑部，需提供信息：

班级、学号及联系方式（QQ 及电话）。会 LaTeX 排版的同学优先。加入编辑部后将统一组织排版、稿件处理规范等工作培训。

审稿人团招募要求

欢迎西安交通大学所有年级（本硕博不限）的同学加入[1]，需提供信息：学校、年级、联系方式（QQ 及电话）、能够审稿的领域，以及本人在该领域的代表性文章一篇（课程论文、毕设论文或正式期刊论文等均可）。有科研工作经历的同学优先。

请有意向者，以「《珠峰学报》招募-部门：姓名」为主题（如「《珠峰学报》招募-编辑部：张三」），将需要提供的信息发至《珠峰学报》官方邮箱：qianyuanxuefu@163.com。

后记

《珠峰学报》为同学们提供了难得的体验期刊工作流程的机会。加入《珠峰学报》，无论是掌握 LaTeX 排版、熟悉审稿流程，还是与作者讨论文章修改意见，都对未来的学术工作大有帮助。

《珠峰学报》诚邀您的加入！

《珠峰学报》愿同您一起成长！

[1]注：《珠峰学报》与大学生基础科学联合学术论坛保持合作。若非西安交通大学学生，也可通过参加该论坛的活动，经审核后加入审稿人团。

文章速览

《珠峰学报》编辑部

该板块是部分作者提供的文章通俗介绍，旨在让读者快速了解文章的目标、结构、思路，快速定位到自己感兴趣的文章详细阅读，并有助于读者更加深入地理解论文，特展示于此。以下材料均来自于论文对应的作者。此外还有很多优质的文章供读者阅读。

1 基于热电比拟的二位热传导虚拟仿真实验的开发

作者：郝瑞霆，应雨倩，刘明阳

本文开发了“热电比拟”虚拟仿真教学实验平台。该平台可作为解决二维热传导问题的通用工具，具备灵活的交互性、丰富的功能、强大的扩展性以及高度的仿真精度等优势，有效帮助突破传统实验对硬件数量和规模的限制。基于热电比拟基本原理，本文推导建立了二维稳态导热物体等效电路，并实现了等温、对流、绝热、界面连续边界条件的求解此外，还添加了节点热平衡方程列写、数据导出等教学辅助功能，有助于加深学生对实验原理的理解。通过将本实验仿真获得的二维墙角导热热平衡偏差与 Fluent 软件进行对比，表明本实验具有较高的准确度。自 2023 年 3 月设计开发完成以来，该虚拟仿真实验已经在本院 2 个班级完成授课，即将面向全校传热学课程实验授课。

2 从言与意的张力谈对《庄子》的阅读体认——以庄惠“濠梁之辩”寓言为例

作者：焦成城

言意关系是庄子哲学的重要议题，通过对《庄子》文本的分析，本篇文章认为庄子并非完全反对一切言语，而是强调语言的界限，突出从“言不尽意”这一现实境遇出发，经由“得意忘言”的实现路径，最终朝向超越“言意”的达道境界这三重逻辑。

砥砺——自然

基于热电比拟的二位热传导虚拟仿真实验的开发

郝瑞霆，应雨倩，刘明阳

摘要

针对热电比拟实验教学环节受限于电路耗材数量和规模导致的实验内容灵活性差、难以拓展问题，基于热电比拟基本原理，建立了含有等温、对流、绝热、界面连续边界条件二维稳态导热物体等效电路，开发了热电比拟虚拟仿真教学实验，实现了本校该教学实验中等温与对流边界的模拟。此外，实验还添加了节点热平衡方程列写、数据导出等教学辅助功能，有助于加深学生对实验原理的理解。将本实验仿真获得的二维墙角导热热平衡偏差与 Fluent 软件进行了对比，表明本实验具有较高的准确度。新开发的虚拟仿真实验自 2023 年 3 月设计开发完成以来，目前已经在本院 2 个班级完成授课，即将面向全校传热学课程实验授课。

关键词：热电比拟　传热学　实验教学　虚拟仿真　导热

1 引言

传热学是能源动力类专业的基础课程，而导热问题是传热学教学中的重点内容之一。对于形状、边界条件复杂的固体导热问题，通常难以求得温度场分布精确的解析解，而将热电比拟法应用于这类问题可以对分析和求解过程进行极大的简化 [1]。热电比拟法利用导热与导电过程的相似性，根据差分方程建立与原物体等效的电路网络，间接求解物体内部温度场。从教学应用角度，传统的热电比拟实验教学在很大程度上受限于实验耗材的数量和规模。将仿真软件与热电比拟原理相结合，则可以摆脱耗材的限制，允许使用者自行设计图形并输入参数。将仿真软件应用到教学中，可以加强学生对热电比拟基本原理的理解和应用，培养学生的研究和探索能力 [2]。

运用热电比拟原理，胡申华 [3] 采用 Excel 电子网格、孟婧 [4] 等使用 Tina-Ⅱ、江文贱 [5] 使用 Multisim 对二维建筑物墙体的等温及对流边界模型进行了数值求解。本质上，只是利用将预先设置好的导热物体等效电路进行电路仿真，并未在软件层面系统实现导热物体向电路系统等效转化，相较于搭建并测量实体电路并未显著提高效率。在软件开发方面，张程宾等应用 MATLAB 开发了导热实验仿真软件，仅限于二维矩形温度场的模拟，能够处理不同边界条件下不含内热源、稳态或非稳态导热问题 [6]；王辉等利用 Fluent 开发了导热虚拟仿真实验平台，仅限于球体一维导热 [7]；梁秀俊等基于 Java 语言开发了具有可视化交互和后处理功能的导热实验虚拟仿真软件，仅包括平壁、圆筒壁、球壳、等截面直肋、二维矩形区域 5 个实验模块 [8]。这些软件所涵盖的仿真模块具有一定的局限性，不能普遍适用于任意边界形状的热传导问题.

本文基于 Visual Studio 2022 平台 MFC 类库，利用 C++ 语言开发了热电比拟虚拟仿真教学实验软件。该软件将本校《热流体课程实验》教材 [12] 热电比拟实验内容转化为仿真软件，并增添了含圆弧、线段和圆形构成的任意形状及多物体接触的二维导热物体温度场求解功能。本文介绍了仿真软件的原理及开发过程，并以实际教学实验中的二维导热问题为例对仿真结果进行了分析。新开发的虚拟仿真实验于 2023 年 3 月设计开发完成，目前已经投入课程使用。

2 软件原理介绍

2.1 传热问题的数值解法

传热问题数值求解的传统思路是将导热物体在时间和空间上连续的温度场用有限个离散点的温度值集合来代替，再通过求解这些值的离散方程得到温度分布的数值解。如图 1 所示，用一系列相交网格线划分求解区域，网格线间的交点称为节点，相邻两节点间的距离称为步长，记为 Δx 、Δy，以节点为中心、由相邻两节点连线中垂线构成的阴影区域称为控制面积。

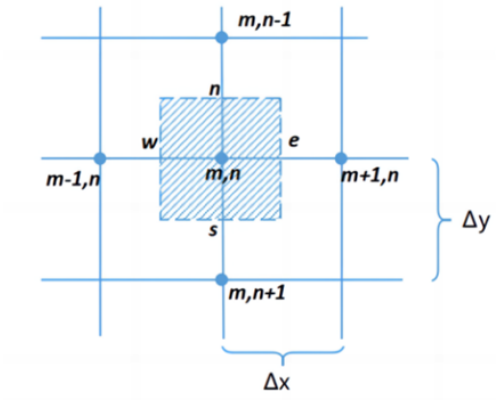


图 1: 网格划分

控制面积完全包含在物体区域内的节点称为内节点 (D)，否则称为边界节点。边界节点根据位置不同，可分为平直边界节点 (B)、内部角点 (E)、外部角点 (ACFGH)。边界节点的控制面积被物体边界截断，截面部分的边长之和称为节点的边界长度，如图 2 所示。

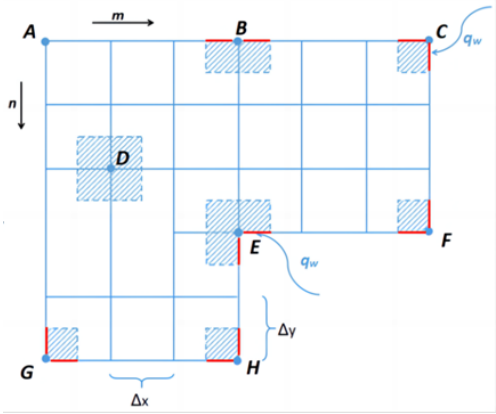


图 2: 各类节点示意图
h

以左上角为原点、m 为横坐标、n 为纵坐标建立节点编号的平面直角坐标系。根据区域离散化和热平衡法，可以得到各节点的离散方程。以内部角点 E 为例，分析有内部节点、边界节点和周围环境向其传热，可得

$$\lambda\frac{t_{m,n-1}-t_{m,n}}{\Delta y}\Delta x+\lambda\frac{t_{m-1,n}-t_{m,n}}{\Delta x}\Delta y+\lambda\frac{t_{m,n+1}-t_{m,n}}{\Delta y}\frac{\Delta x}{2}+\lambda\frac{t_{m+1,n}-t_{m,n}}{\Delta x}\frac{\Delta y}{2}+q_{wx}\frac{\Delta y}{2}+q_{wy}\frac{\Delta x}{2}=0 \tag{1}$$

式中：t 为节点温度 /°C；q_w 为热流密度，$\mathrm{W/m^2}$；λ 为物体导热系数，$\mathrm{W/(m\cdot K)}$。

2.2 热电比拟原理及扩展

对于电路，仅考虑含电阻、独立电压源、独立电流源三种元件，根据基尔霍夫电流定律、基尔霍夫电压定律和欧姆定律，可得节点电压方程的一般形式：

$$AYA^TU_n=AYU_s-AI_s \tag{2}$$

式中：A 为关联矩阵，表示节点与支路的拓扑结构关系；U_n 为节点电压矩阵，V；U_n 为支路独立电压源值，V；I_s 为支路独立电流源值；Y 为导纳矩阵，Ω^{-1}，$Y=diag(1/R,...)$。

当满足条件：电路的拓扑结构信息即每个支路的始末节点编号已知，可列写关联矩阵；公式 (2) 中的电导值 Y、电压源值 U_s、电流源值 I_s 已知时，能够解得节点电压值 U_n。热电比拟的基本原理是电势场、温度场在二维平面上均满足拉普拉斯方程，因此可用电压 (V) 比拟节点温度 (°C)、电流 (A) 比拟热流 (W/m)、电阻 (Ω) 比拟热阻 (m • K/W)，通过列写节点电压方程，求解导热物体的等效电路电压分布，间接求解温度场分布。

根据热电比拟原理，设定步长 $\Delta x=\Delta y$，对节点离散方程、边界条件进行处理，得到等效电阻值、电压源值或电流源值。

(1) 内节点

$$\sum_i^4\frac{t_i-t_{m,n}}{1/\lambda}=0=\sum_i^4\frac{t_i-t_{m,n}}{R_{Ii}} \tag{3}$$

式中：R_I 为内部热阻 /Ω；i 为相邻节点编号。

(2) 边界节点 (等温、对流、绝热边界条件)

$$\begin{aligned}\sum_i\frac{t_i-t_{m,n}}{1/\lambda}+\sum_j^2\frac{t_j-t_{m,n}}{2/\lambda}+\sum_k q_{wk}l_k=0\\=\sum_i\frac{t_i-t_{m,n}}{R_{Ii}}+\sum_j^2\frac{t_j-t_{m,n}}{R_{Bi}}+\sum_k I_{con}\end{aligned} \tag{4}$$

式中：q_w 为热流密度，$\mathrm{W/m^2}$；l 为边界长度，m；R_B 为边界热阻，Ω；I_{con} 为边界热流项，A；i 为相邻内节点编号；j 为相邻边界节点编号；k 为边界节

点所在的边界编号。

对于对流边界，设 t_0 为环境温度，/°C；h 为换热系数，W/(m^2 · K)。则

$$q_w = \frac{t_0 - t_{m,n}}{1/h} = \frac{t_0 - t_{m,n}}{R_C} \tag{5}$$

式中：R_C 为对流热阻，Ω。特别地，$h \to \infty$ 等效于恒温边界，$h \to 0$ 等效于绝热边界。

(3) 界面连续边界条件 (多物体接触)

设两种材料接触良好，满足接触节点的温度值相同、热流相同，则两连续边界可等效为并联电路，如图 3 所示，其等效热阻为

$$R_B = \frac{R_{B1}R_{B2}}{R_{B1} + R_{B2}} = \frac{2}{\lambda_1 + \lambda_2} \tag{6}$$

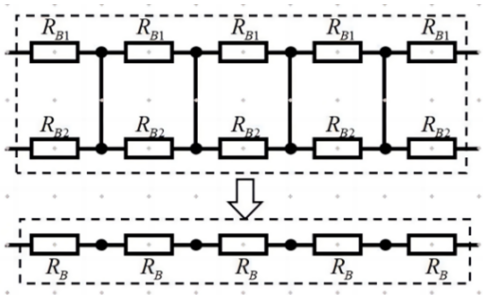


图 3: 界面连续边界条件等效电路

综上可得，在任意边界条件下，任意节点的统一等效形式。二维导热物体等效电路结构如图 4所示，元件数值计算公式如表 1 所示。

$$\sum_i \frac{t_i - t_{m,n}}{1/\lambda} + \sum_j^2 \frac{t_j - t_{m,n}}{2/\lambda} + \sum_k \frac{t_0 - t_{m,n}}{1/hl_k} = 0 \tag{7}$$

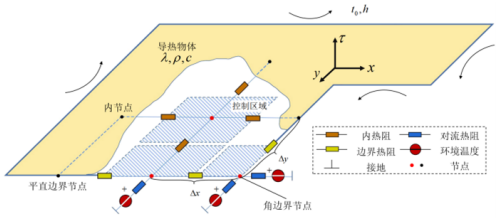


图 4: 热电比拟基本原理图

表 1: 支路原件数值计算公式

支路类型	电阻值	电压源值	电流源值
内部支路	$R_1 = 1/\lambda$	0	0
边界支路	$R_B = 2/(\lambda_1 + \lambda_2)$	0	0
对流支路	$R_C = 1/hl$	$U_C = t_0$	0
	0	0	$I_{con} = q_w l$

3 软件开发

3.1 软件设计模式及开发框架

软件主要采用 MVC 设计模式，包括分离的数据模型处理系统和图形界面交互系统。在每个功能模块的实现部分还进行了分层设计，以降低模块间的依赖性。将复杂的功能模块拆分为若干流程进行模块化编程，并在不同模块之间采用接口相连，从而支持子模块对程序进行调试、修改与功能拓展。软件主界面如图 5 所示，由菜单栏、工具栏和绘图界面构成。

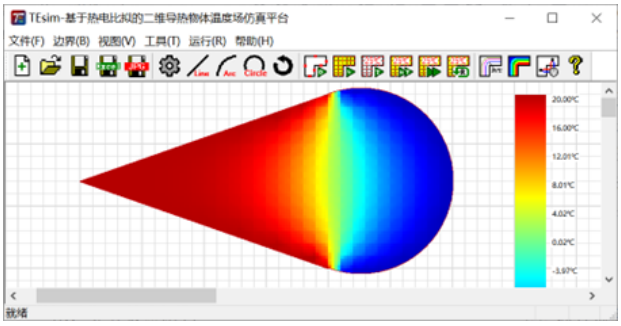


图 5: 软件主界面

3.2 软件流程设计

软件的基本流程如图 6 所示：

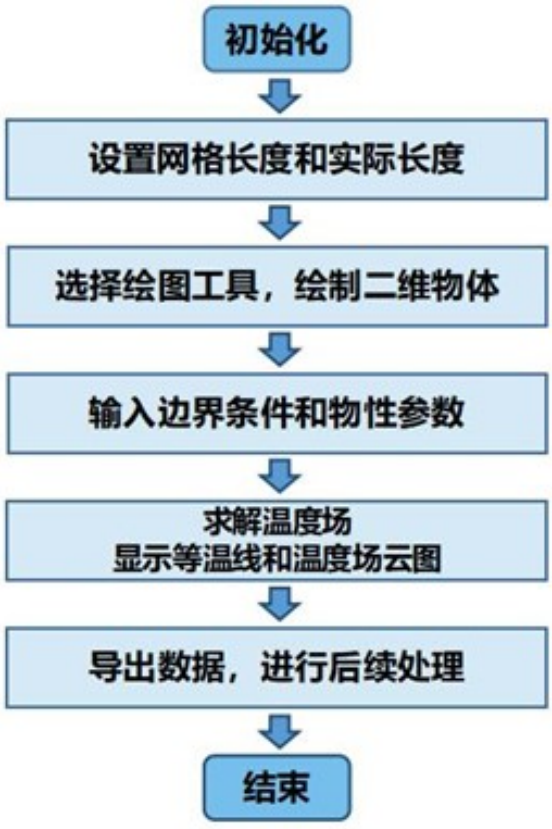


图 6: 软件基本流程

(1) 设置网格像素长度及对应的实际物体长度；

(2) 选择直线、圆弧、圆形三种基本绘图工具首尾相连绘制二维物体，物体边界必须构成封闭图形，选择边界条件类型并输入适当的物性参数，包括恒温、对流、绝热、界面连续 4 种边界类型和环境温度、换热系数、导热系数 3 种参数，完成后提交边界。程序将加载物体封闭区域，判断相互包含关系，建立边界与物体间相互联系，判断边界相互接触关系，即生成界面连续边界条件；若边界形状不封闭或边界条件参数设置不合理，将弹出警告消息等待修改；

(3) 边界设置合理后，用户单击物体区域输入适当的物性参数，若参数设置合理则生成节点；程序将进行边界阶梯化，识别并生成内节点与边界节点，根据节点种类及位置关系建立内部支路、边界支路、对流支路；

(4) 求解温度场，程序将根据物性参数为支路数值信息赋值，根据电路拓扑结构和支路数值信息列写节点电压矩阵方程，使用 *LU* 分解进行直接求解；

(5) 查看温度场求解结果，设置显示所需等温线和温度场云图，程序将根据线性插值算法进行图像绘制；

(6) 导出数据，进行后续处理，或查看节点网格视图、支路数值信息等，完成后返回绘制边界的环节重新开始。

4 软件功能

4.1 二维墙角对流传热问题仿真

以本校《热流体课程实验》教材 [12] 热电比拟实验中二维墙角问题作为仿真案例，如图 7 所示。其中，砖墙外表面与流体的对流换热系数 h_1=10.6W/(m^2 · K)，流体温度 t_1=30 °C，内部为空腔，内表面与流体的对流换热系数 h_2=3.975 W/(m^2 · K)，流体温度 t2=10 °C，物体导热系数 λ=0.53 W/(m • K)。

根据软件使用流程和示例物体的对称性，单击“设置”，设置节点步长为 0.1 m，如图 8(a)；绘制 1/4 墙角如图 8 (d)；单击边界小圆圈输入前述边界条件，如图 8(b)；提交物体后单击物体区域输入物性参数，如图 8(c)；求解温度场并进行可视化处理，最终得到如图 8(e) 的温度场云图与如图 8(f) 的等温线结果。

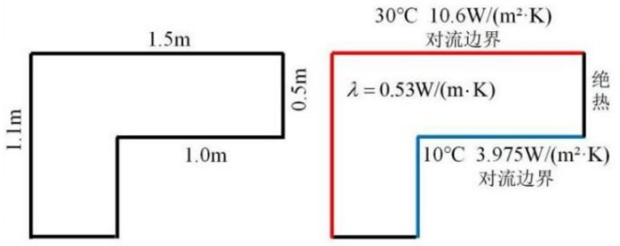


图 7: 二维墙角导热模型

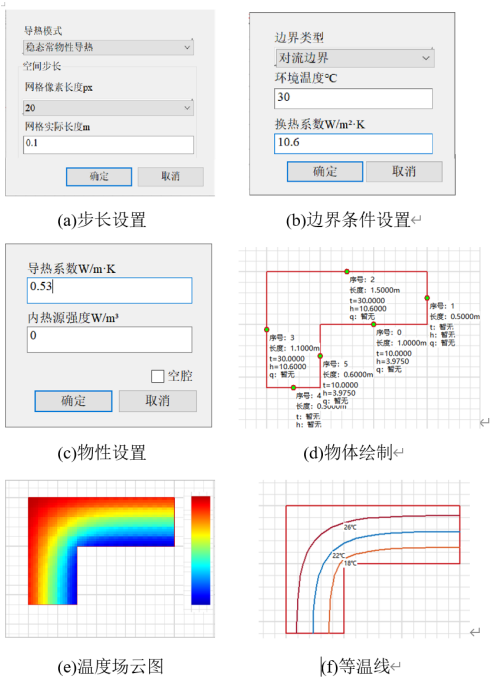


图 8: 参数输入及结果输出界面

4.2 仿真结果验证

以 Excel 格式导出各节点温度值计算结果。为模拟实际实验中万用表的测量精度，将温度值保留四位小数。分别计算得每米高砖墙内表面和外表面的导热量以及热平衡偏差得

$$\Phi_{in} = h_2 \sum_i A_i(t_1 - t_2) = 28.36093\text{W}$$

$$\Phi_{out} = h_1 \sum_j A_j(t_1 - t_2) = 28.36094\text{W}$$

$$\Delta\Phi = \frac{|\Phi_{out} - \Phi_{in}|}{\frac{\Phi_{out} + \Phi_{in}}{2}} \times 100\% \approx 2.0 \times 10^{-5}\%$$

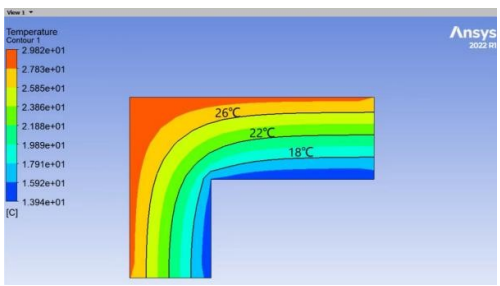


图 9: Ansys Fluent 软件仿真结果

4.3 拓展及教学辅助功能

热电比拟实验实物电路存在难以拓展的问题，仅能针对单一导热问题重复测量。如图 5 所示，虚拟仿真实验能够绘制任意直线、圆弧形状首尾相连的物体，同时方便调节边界条件、物体物性、节点步长参数，有助于在测量性实验基础上开展探究性实验，极大地丰富了实验内容。

为了在虚拟仿真教学过程中体现热电比拟基本原理，避免“黑箱”实验，软件添加了节点网格视图作为教学辅助功能。节点网格视图是网格划分后显示节点及支路位置分布的视图，显示了导热物体等效电路结构，与实物电路相一致，如图 10 所示。

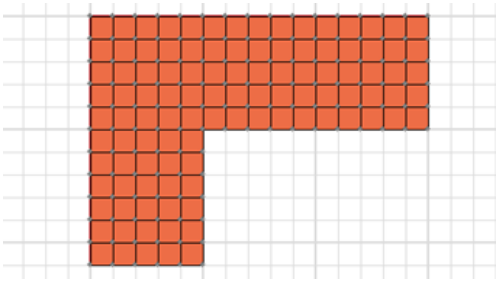


图 10: 节点网格视图

使用鼠标左键单击节点，软件将弹出对话框并输出节点类型、边界长度、控制面积大小、温度值，以及所有相连支路的类型、电阻值、电压源值、电流源值，如图 11 所示。根据以上数值信息，学生能够列写节点热平衡方程，从中强化对实验原理的理解，学习热传导问题数值解法的有关知识。

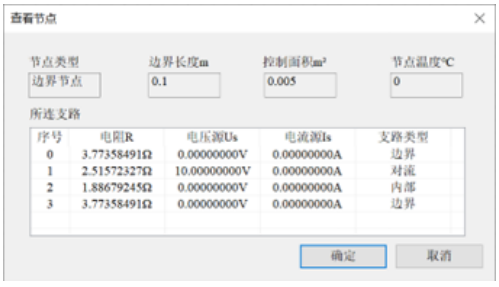
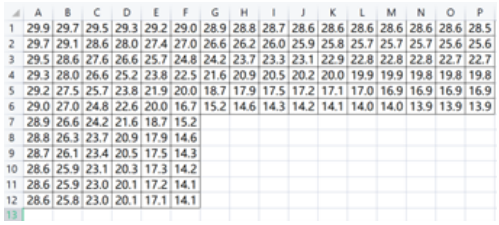


图 11: 节点数值信息

为了便于学生进行数据处理，软件还支持将计算后得到的节点温度数据导出为 Excel 表格，根据节点坐标位置输出至指定单元格中，最高可精确至小数点后四位。如图 12 所示。



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	29.9	29.7	29.5	29.3	29.2	29.0	28.9	28.8	28.7	28.6	28.6	28.6	28.6	28.6	28.6	28.5
2	29.7	29.1	28.6	28.0	27.4	27.0	26.6	26.2	26.0	25.9	25.8	25.7	25.7	25.7	25.6	25.6
3	29.5	28.6	27.6	26.6	25.7	24.8	24.2	23.7	23.3	23.1	22.9	22.8	22.8	22.8	22.7	22.7
4	29.3	28.0	26.6	25.2	23.8	22.5	21.6	20.9	20.5	20.2	20.0	19.9	19.9	19.8	19.8	19.8
5	29.2	27.5	25.7	23.8	21.9	20.0	18.7	17.9	17.5	17.2	17.1	17.0	16.9	16.9	16.9	16.9
6	29.0	27.0	24.8	22.6	20.0	16.7	15.2	14.6	14.3	14.2	14.1	14.0	14.0	13.9	13.9	13.9
7	28.9	26.6	24.2	21.6	18.7	15.2										
8	28.8	26.3	23.7	20.9	17.9	14.6										
9	28.7	26.1	23.4	20.5	17.5	14.3										
10	28.6	25.9	23.1	20.3	17.3	14.2										
11	28.6	25.9	23.0	20.1	17.2	14.1										
12	28.6	25.8	23.0	20.1	17.1	14.1										
13																

图 12: 节点温度值导出

5 结语

基于软件灵活设置物体形状和实验参数的特点，虚拟仿真实验教学将有助于学生深入探索传热过程的影响因素和基本原理，弥补传统教学中实际操作的不足，达到提高学生学习兴趣与自主探究能力的目的[14][15]。本文基于热电比拟原理开发出虚拟仿真教学实验，实现了在等温、对流、绝热、界面连续边界条件下，任意形状二维物体稳态导热过程的数值模拟和结果可视化演示，具有操作简单、准确度高、可视化效果强等优点。本实验已于 2022-2023 学年第二学期在本校《热流体课程实验》中面向智能钱 001、能动 B003 班 55 名学生完成试运行，于 2023-2024 学年第一学期正式投入传热学实验课程使用，作为“二维导热物体温度场电模拟实验”线上虚拟仿真教学实验，并且在“导热系数测量实验”中间的 2 小时等待时间内进行该实验，以充实导热系数测量实验的内容。

参考文献

[1] 程瑞, 王馨, 张寅平. 基于热电比拟的建筑外墙热性能分析方法 [J]. 工程热物理学报, 2014, 35(05): 978-981. ［C］, v

[2] 朱赤, 幸文婷, 叶晓明. 空气纵掠热板的边界层测定实验设计与教学 [J]. 实验室研究与探索, 2022, 41(12): 244-247.

[3] 胡申华. 基于 Excel 的模拟墙体导热问题分析 [J]. 实验研究与探索, 2012, 31(10): 53-55+108.

[4] 孟婧, 张可, 魏旸, 等. Tina-Ⅱ在“传热学”热电比拟实验中的应用 [J]. 实验室研究与探索, 2021, 40 (9): 114-118.

[5] 江文贱, 徐隽. Multisim 在《传热学》中的应用 [J]. 江西电力职业技术学院学报, 2008, (01): 58-59.

[6] 张程宾, 韩群, 陈永平. 基于 MATLAB 的传热学课程虚拟仿真实验平台设计 [J]. 实验技术与管理, 2020, 37 (1): 132-136.

[7] 王辉, 解明杰, 向文凤, 等. 基于 Fluent 的传热学虚拟实验平台的开发 [J]. 大学教育, 2019, 8(3): 80-83.

[8] 梁秀俊, 刘璐, 刘彦丰, 等. 基于 Java 语言的在线导热实验虚拟仿真软件开发 [J]. 实验室研究与探索, 2022 (02): 106-110.

[9] 钟建华. 基于 Qt 的电路仿真软件开发 [D]. 杭州: 杭州电子科技大学, 2021.

[10] 陶文铨. 传热学 [M]. 北京: 高等教育出版社, 2018.

[11] 陶文铨. 数值传热学 [M]. 西安: 西安交通大学出版社, 2001.

[12] 王小丹, 孟婧, 张可, 等. 热与流体实验教程 [M]. 西安: 西安交通大学出版社, 2017.

[13] 黄晓齐. 有内热源时稳定导热的电模拟计算 [J]. 贵州工学院报, 1984(Z1): 13-25.

[14] 曹玮, 许亚敏, 陆紫生. 能源动力类虚拟仿真综合实践平台建设 [J]. 中国教育信息化, 2021(04): 71-74.

[15] 戴前进, 刘强, 张学杨, 等. 新工科背景下实验-理论-仿真在“传热学”第二课堂的实践 [J]. 装备制造技术, 2021(4): 196-199.

[16] 李澜, 王吉. 高等学校虚拟仿真实验教学现状及趋势研究 [J]. 中国教育技术装备, 2022(19): 18-21+25.

作者信息

郝瑞霆，应雨倩，刘明阳 西安交通大学能源与动力工程学院 陕西 西安 710049

Fast And Comprehensive Dependency Scanner Using Clang

Junyang Zhang

图 1: The Bytedance office building in Beijing, 2022.

摘要

Dependency scanning of C/C++ projects can be acheived by using the compiler's preprocessor options. However, turning on these options will require the full execution of the frontend, which is time consuming for large projects consisting of hundreds of thousands lines of codes. In this paper, a fast dependency scanner using the features of the LLVM-Clang 15 frontend is proposed. By bypassing the syntax actions and reconstructing the preprocessor, our scanner can acheive more than 50x speed advantage compared to traditional methods. We also constructed a production-ready development tool to demonstrate the performace of our scanner in industrial scenarios.

关键词： *compiler frontend* *preprocessor* *dependency* *scanning* *llvm-clang*

1 Introduction

Despite the introduction of modules in the C++20 standard, the most common and suggested way to share code between compilation units in C/C++ projects is to use the `#include` preprocessor directive[1]. When using the include directive in a C++ file, it is effectively the same as replacing the directive with the full content of the included file. Due to its simple behavior, the processing of the include directives can be done without semantic analysis, and it is often delt by the preprocessor. However, the simplicity causes some problems since it allows the headers to recursively include each other. The size of the compilation units tend to grow faster than the size of the project even if the code are reasonably seperated in many source files and headers. The behavior of the include directive leads to long compile time of large projects and also long processing time of various development tools, such as Doxygen[2] and clang-tidy[3].

From the software configuration management perspective, there is a need of sorting out the dependencies of a project when it contains over ten thousands of lines of code. Many development tools such as Microsoft Visual Studio and Doxygen included features such as dependency view and XML generation. These features relies on the preprocessor options of the compiler. For example, using `-E -M` options of GCC can produce the full dependency of a file[4]. While the `-E` option tells the compiler to stop after the preprocessor, the compiler is not

smart enough to bypass the time consuming lexical analysis of the preprocessor. This problem is addressed by Lorencz et al.[5] and handled in the LLVM version 15 release. However, the new implementation, clang-scan-deps lacks a important feature. It makes dependency scan fast by bypasssing the lexical analysis of the file except the preprocessor directives. But it won't expand the include directives to fulfill the recursive inclusion feature, thus tooling develpers must implement their own recursive inclusion scanning logic. Considering the limits of clang-scan-deps, we present a dependency scanner that can handle recursive inclusion, while much faster than vanilla approaches.

This work is produced during the author's internship at Bytedance Inc. All materials shown in this paper is desensitized and shall not have public usage.

2 Overview

In this paper, we firstly introduce Clang and the Clang LibTooling API, which is the basis of our scanner. Secondly we present our ideas of constructing the new scanner. Thirdly we make comparisons of our scanner and the vanilla approaches in the terms of speed and accuracy. Finally we briefly showcase our Proto Auto Cut tool as a case of actual application of our scanner.

The following reference of Clang or LLVM components is under the LLVM 15 context, otherwise noticed. The features that this paper mentions may change in future stable releases.

3 Clang LibTooling API

Clang is a project under the LLVM monorepository, which provide the frontend for C, C++ and Objective-C languages. Clang is open, modular and effective. Utilizing the power of LLVM, clang can basicly outperform gcc as a generic compiler.

LibTooling is a library to support writing standalone tools based on Clang[6]. The LibTooling API is built around the concept of "`FrontendAction`". Using a "`ClangTool`" class instance, one can run any predefined or custom frotend actions over code. Also, the standalone tool can be linked with any LLVM or Clang libraries including AST, driver, rewriter, static analyzer, compilation database, etc. Clang also provided common tools like clang-format, clang-fuzzer, clang-tidy, which is developed upon the tooling APIs. Our scanner is a standalone tool using the LibTooling API and other necessary Clang APIs.

However, not everything is exposed through Clang APIs, such as "`HeaderSearch`", which is essental in finding headers using the directory arguments and default paths. Handing header searching without any APIs would cause different behaviors of the tool and actual build, thus making the tool unreliable. We adopted to copy the source code of the Clang preprocessor implementation and make necessary refactors to ensure consistency in header searching order, which will be discussed later.

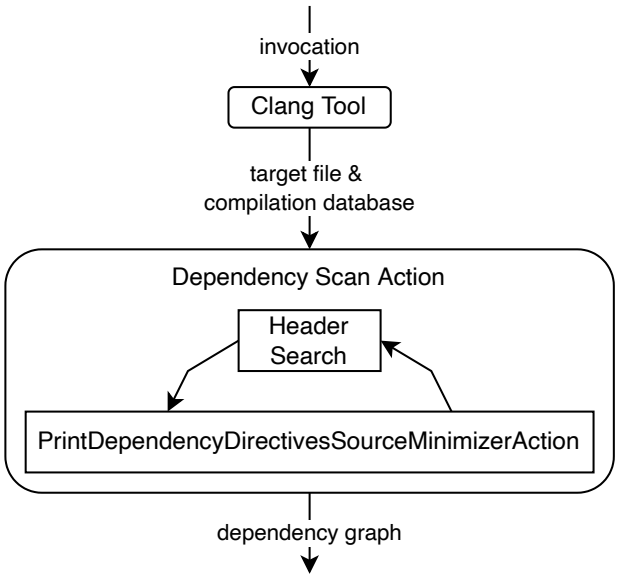


图 2: The scanner architecture.

4 Scanner Architecture

In this section we describe our implementation of our scanner. Figure 2 shows the architecture of our scanner in the aspect of the execution order. The scanner has an instance of `ClangTool` and can be invoked through the command line. The `ClangTool` will run a "Dependency Scan Action" over the designated file, and produce a dependency graph. We will explain the details in this order.

4.1 Invocation

The scanner can be invoked in two ways: 1. with a compiler invocation of a single file; 2. with a compilation database. The first way requires the full arguments to compile the file, for the scanner to properly find the headers and opt the designated standard of the language. With the first way, only the dependency of this single file will be found. The second way is use to scann all the dependencies in a single project. And the second way requires a compilation database.

The compilation database is a JSON format file containing the information of a build. Build systems such as CMake[7] and Bazel[8] runs a set of commands to process, compile and link the whole project. Such build system can be configured to write a compilation database to record the build. And Clang has a API for accessing the compilation database to analyze the build, which is used in our work. With the compilation database, the scanner can be run multiple times over all the files of the project and avoid unnecessary filesystem invocations.

Next, the instance of `ClangTool` will process the invocation and run the customized `DependencyScanAction` over the file. While running the action, not only the target file and the database could be accessed, but also a bunch of Clang class instances including the driver and the compiler instance.

4.2 Dependency Scan Action

This action is the key part of our scanner, which includes the header search and a scanning action. Header searching begins when a valid include directive is read by the preprocessor, and the preprocessor raises a preprocessor callback. The registered header scan instance then searches for headers sequentially from a list of directories or checks, which contains:

(1) the current path of the includer;
(2) the suggested module if the includer is a system header;
(3) user designated paths using the commandline;
(4) system default paths, usually set by the gcc driver;

The PrintDependencyDirectiveSourceMinimizerAction is also a key part of the dependency scan action. This lengthy-named action ls introduced in LLVM 15, which is the basis of the clang-scan-deps tool. This action will read but not tokenize the input stream, consuming only the preprocessor directives. Listing1 is an example of a scanned source file, in which the highlighted lines are the lines that is processed by the action. Since the amount of directives in a file is usually small compared to other code, this action can speedup the preprocessing phase by a large fraction.

The action nontheless cannot recursively process the included headers, thus it needs to do header search when it is necessary. Fortunately the action can have regular preprocessor callbacks, thus we shall register the header search function as the callback. In the perspective of control flow, the action and the header search function will repeatedly call each other until the last header is processed.

With the completion of the preprocessing, the dependency of the file is available. Due to the behavior of the include directive, the output should be a directed graph with nodes representing files and edges representing inclusions.

5 Implementation and Evaluation

We implement the scanner as a standalone tool and tested over some internal projects at bytedance. The testing environment is a 96 core intel physical server equipped with 1T DDR4 memory, and we use 32 cores locked at 2.5 GHz. The operating system is Debian 10 buster with Linux kernel 5.15.86. We use llvm 11 toolchain to compile all the tools that are used in the evaluation, and we link our scanner with LLVM 15.0.3 libraries.

To evaluate the scanner we take our target project and build it using bazel, the build system, for the compilation database. And then use the compilation database as the input for the scanner to scan the project. We compare the performace of the scanner with both gcc 8.3.0 and clang 11 solutions. The solution using these compilers is a python script that pipes the output of each command in the compila-

tion database, appended with "`-E -M`" options. The script will record the processing time of the subprocesses as the execution time, instead of the total completion time.

We choose 4 typical large C/C++ project as the target project for evaluation, and their characteristics are listed in Table 1. $Pred_A$, $Pred_B$ and Parse are internal projects of Bytedance thus we use pseudonym here. These projects can represent actual industrial use cases.

Project	LoC	Description
Linux	~28 million	For reference
$Pred_A$	~17 million	A datacenter application for prediction
$pred_B$	~2 million	A datacenter application for prediction
Parse	~16 thousand	A parsing library

表 1: Characteristics of the target projects.

Figure 3 shows the result of our experiment using our scanner and the selected projects. For each project, the processing time of the scanner and the compiler scanning solutions are recorded, along with the compilation time using clang 11 for comparison. The result meets our expectation that our dependency scanning time is significantly lower than that of the compiler solutions, and is a fraction of the full compilation time since most procedures such as syntax analysis, code generation and optimization are ignored. Our scanner outperform general solutions by around 6x, bringing significant improvement to the development tools that need dependency scanning.

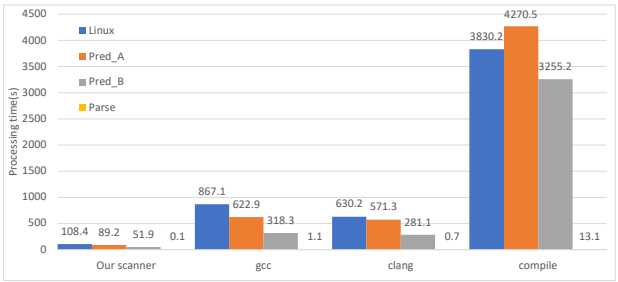


图 3: The result of the experiment.

For the accuracy part of the evaluation, our test examples covered system libraries, `#include_next` directives, sophisticated compiler options, custom default paths and preprocessor macros. We conclude that the output of our scanner is topologically identical with the general solutions.

The compatibility of our scanner is good, for it only needs the compilation database, which is supported by most of the build systems. However, configurations or scripting jobs are needed when handling GNU Make projects like Linux. The automation of our tool is even better than the general dependency scanning tools in the terms of user experience.

6 A Use Case

In this section a use case of the scanner is proposed, to elaborate the significance of a fast scanner in industrial applications. We designed Proto Auto Cut, which is an automation tool to accelerate compilation for large projects using protocol buffers[9].

Protocol buffers is a mechanism for serializing structured data, with similar goal as XML. But protocol buffers encode data into byte streams rather than text streams, making it smaller, faster and efficient, only compromising human readability. To use protocol buffers, one should install the library for the language the user selected, and write a ".pb" file containing the classes that need to be serialized, and the instances of the classes are called "messages". The ".pb" file is written using protocol buffer's interface description language (IDL).

Protocol buffers is widely used at Bytedance, and some legacy code contains discouraged usage of protocol buffers. There are some projects sharing a ".pb" file which has a super long message consisting of nearly ten thousand fields. The problem then raises. The C++ layer for protocol buffers translate the ".pb" file into C++ classes, an a super long message will lead to a parsing function containing a switch argument of ten thousand cases. Such a C++ function stressed the compiler's register allocation pass and prolonged the file's compilation time to nearly 40 minutes.

The way we takle this problem is to shrink the size of the ".pb" file. Because the number of the fields that the projects is using is around 20, far less then that in the ".pb" file, and protocol buffers allowed undefined fields in the wire, we can remove the fields that are not used in the project. We developed an auto tool for this shrinking process, the Proto Auto Cut. Using our dependency scanner, Proto Auto Cut can tell the files that referenced generated protocol buffer headers, and do semantic analysis over these

files to extract the fields that are actually used. With the power of the `protoc` library, we can remove the unnecessary fields in the file and do the compilation.

In practice, the Proto Auto Cut tool boost the compilation speed of the legacy projects up to 60x, which massively reduced the workflow of the develpers and loosen the stress on our compilation cluster. The dependency scanner in this case reduces the processing time of our tool by 5x, making it possible to be merged into our build process.

参考文献

[1] c++2011iso

[2] van2008doxygen

[3] sadowski2014usable

[4] gcc2023options

[5] lorencz2018clang

[6] clang2023libtooling

[7] bast2018cmake

[8] maudoux2018correct

[9] kaur2010evaluation

Efficient Embeddings of Logical Variables for Query Answering over Incomplete Knowledge Graphs

Dingmin Wang, Yeyuan Chen, Bernardo Cuenca Grau

摘要

The problem of answering complex First-order Logic queries over incomplete knowledge graphs is receiving growing attention in the literature. A promising recent approach to this problem has been to exploit neural link predictors, which can be effective in identifying individual missing triples in the incomplete graph, in order to efficiently answer complex queries. A crucial advantage of this approach over other methods is that it does not require example answers to complex queries for training, as it relies only on the availability of a trained link predictor for the knowledge graph at hand. This approach, however, can be computationally expensive during inference, and cannot deal with queries involving negation.

In this paper, we propose a novel approach that addresses all of these limitations. Experiments on established benchmark datasets demonstrate that our approach offers superior performance while significantly reducing inference times.

1 Introduction

Knowledge graphs (KGs) are graph-structured knowledge bases where nodes and edges represent entities of interest and their relations (Hogan et al.2021; Heist et al.2020). Formally, a KG can be seen as a set of triples (or logical facts) and can be represented in standard formats such as the Resource Description Framework (RDF). KGs are increasingly been used in a wide range of applications, such as Web search, question answering in digital assistants, recommender system, scientific discovery, or data integration (Li et al. 2020; Xu et al. 2020; Noy et al. 2019). Many largescale KGs used in applications, however, are highly incom- plete; as a result, *knowledge graph completion*—the problem of completing a KG with missing facts that are likely to hold in the domain of interest—has received a great deal of attention in recent years (Rossi et al. 2021).

Link predictors are ML models that can predict whether individual missing facts hold in an incomplete KG. These models can be very effective for KG completion tasks. They typically work by first learning vector representations of the entities and relations in the KG during training, and then exploiting these embeddings to predict individual facts.

In recent years, however, there has been growing interest in going beyond the prediction of simple logical facts and tackle the more general problem of answering complex FOL queries over incomplete KGs. Roughly speaking, given a query $Q(x)$ with answer variables x and a KG $\mathcal{K}$, the goal is to compute the answers to $Q(x)$ with respect to the (unknown) completion $\mathcal{K}^*$ of $\mathcal{K}$. One possible solution to this problem is to exploit the availability of a neural link predictor to first compute the complete KG $\mathcal{K}^*$ and then evaluate $Q(x)$ directly over $\mathcal{K}^*$ using standard query answering technology in data management. This solution, however, can be problematic in practice as it may require an unmanageable number of link prediction tests and may involve the materialisation of a large number of facts in the KG.

As a result, research has focused on techniques for answering queries without the need of complet-

ing the graph first. *Query embedding methods*, such as box embeddings (Ren, Hu, and Leskovec 2020), beta embeddings (Ren and Leskovec 2020), and cone embeddings (Zhang et al.2021), compute vector embeddings for the queries and then treat query answering as a nearest-neighbor search problem in the latent space. Training such models is, however, problematic, as they can only deal with query patterns seen during training, and hence training typically requires the availability of answers to a wide variety of queries.

To address this limitation, Arakelyan et al. proposed the CQD model, which relies only on the availability of vector embeddings provided by a trained neural link predictor and hence does not require example answers to complex queries for training (Arakelyan et al.2021). This approach, however, also comes with a number of limitations. First, it is restricted to a subclass of positive existential queries—FOL queries constructed using only conjunction, disjunction and existential quantification; thus, it cannot handle queries involving other important constructs such as negation. Second, CQD relies on a computationally expensive combinatorial optimisation problem to search for likely query answers using the embeddings provided by the neural link predictor; this can lead to slow inference times and limit the applicability of CQD to scenarios with low latency requirements.

In this paper, we propose a novel approach to query answering over incomplete KGs that addresses these limitations. Our approach extends the query language of CQD to support negation while at the same time significantly reducing the computational resources needed for both the training and inference stages. Furthermore, similarly to CQD and in contrast to query embedding methods, our approach relies solely on the availability of a trained neural link predictor and hence does not require example answers to complex queries for training. We have implemented our approach in a system called Var2Vec. Our extensive experiments on three standard benchmark datasets show that Var2Vec can deliver superior performance while significantly improving both training and inference times.(1)

(1)Code available at https://github.com/wdimmy/Var2Vec.

2 Background

Knowledge Graphs.A knowledge graph $\mathcal{K}$ over a vocabulary consisting of pairwise disjoint sets of entities ε and relations $\mathcal{R}$ is a finite set of triples $\langle e_1, r, e_2\rangle$ where $e_1, e_2 \in \mathcal{E}$ and $r \in \mathcal{R}$.A triple $\langle e_1, r, e_2\rangle$ is equivalent to a fact $r(e_1, e_2)$, and hence a KG $\mathcal{R}$ can also be equivalently seen a FOL knowledge base consisting of a finite set of facts.

Queries. We consider the subclass of FOL queries defined next. Consider a vocabulary consisting of entities $\mathcal{E}$ and relations $\mathcal{R}$,and let $\mathcal{V}$ be a set of variables pair-wise disjoint with $\mathcal{E}$ and $\mathcal{R}$. A term is either an entity in $\mathcal{E}$ or a variable in $\mathcal{V}$. An atom is a formula $r(t_1, t_2)$ where $r \in \mathcal{R}$ and each t_i is a term. A literal is an atom or the negation of an atom. The subclass of FOL formulasφ that we consider is then inductively defined according to the following grammar, where l is a literal and x is a variable:

$$\varphi ::= l|\varphi \wedge \varphi|\varphi \vee \varphi \mid \exists x.\varphi. \tag{1}$$

Variable occurrences in the scope of a quantifier are bound, whereas other variable occurrences are free. A query is a formula with a single free variable, called the answer variable.

Note that we consider an extension of the class of monadic positive existential queries (i.e.,FOL queries with a single answer variable constructed using only conjunction, disjunction and existential quantification) where, in addition, we allow for a restricted form of negation occurring only in front of atoms. This is in contrast to previous works, in which queries were typically restricted to monadic positive existential queries in disjunctive normal form.

The dependency graph of a query $Q(x)$ is a directed graph where the nodes are the terms occurring in the query and where there is a directed edge from term t1 to term t_2 whenever an atom of the form $r(t_1, t_2)$ occurs as a sub-formula in the query. Following the literature, we restrict ourselves to queries where the dependency graph is connected and acyclic, and where the answer variable is the single sink node and each source node in the graph

is an entity (and thus not a variable). In what follows, we will refer to the FOL queries satisfying all the aforementioned syntactic restrictions as admissible. To the best of our knowledge, the class of admissible queries covers all types of queries supported by existing embedding-based query answering approaches in the literature, as well as all types of queries in existing benchmarks. An example admissible query and its corresponding dependency graph are shown in Figure 1.

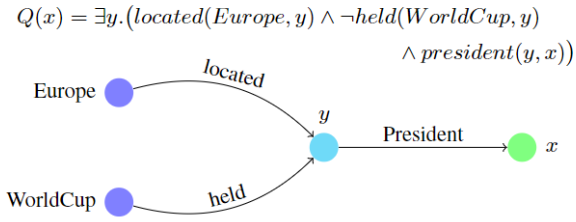


图 1: Formalisation of the query "*list the presidents of European countries that have never held the World Cup*" and the corresponding dependency graph. The sink node corresponding to the answer variable is depicted in green, whereas the source nodes (also called anchor nodes in the literature) are entities depicted in purple.

The semantics of query answering is standard. We say that an entity $e \in \mathcal{E}$ is an answer to query $Q(x)$ with respect to a KG $\mathcal{K}$ if $\mathcal{K} \models Q(e)$, and we denote the set of answers to Q with respect to $\mathcal{K}$ as $[Q]_{\mathcal{K}}$.

Neural Link Predictors. In the context of this paper, we define a neural link predictor M for a finite vocabulary of entities E and relations R as a pair enc_M,dec_M consisting ofan encoding function enc_M and a decoding function dec_M,

respectively. The encoding function maps each entity $e\mathcal{E}$ to a k-dimensional real-valued vector and each relation $r\mathcal{R}$ to a $k^{'}$-dimensional real-valued vector, where k and $k^{'}$ are hyper-parameters of the model representing the dimensions of entity and relation embeddings, respectively. In turn, the decoding function $dec_{\mathcal{M}}$ is a function mapping each triple of vectors in $\mathbb{R}^k \times \mathbb{R}^{k^{'}} \times \mathbb{R}^k$ to a real value between 0 and 1. The vector components in the encoding of each entity and relation of the vocabulary constitute the learnable parameters of the model, whereas the decoding function is used to assign to each possible fact $r(e_1, e_2)$ in the vocabulary a probability $\text{dec}_{\mathcal{M}}$ $(enc_{\mathcal{M}}(e_1), enc_{\mathcal{M}}(r), enc_{\mathcal{M}}(e_2))$.

t-norms and t-conorms. Triangular norms (or $t-norms$) are generalisations of the logical conjunction operator that are typically adopted in fuzzy logics (Héajek 2013; Gupta and Qi 1991). Formally, a t-norm is a binary operation $\top$ on [0, 1] satisfying the following properties:

- $\top(x, y) = \top(y, x)(commutativity)$;
- $\top(x, \top(y, z)) = \top(\top(x, y), z)(associativity)$;
- $y \leqslant z \rightarrow \top(x, y) \leq \top(x, z)(monotonicity)$; and
- $\top(x, 1) = x$ (neutral element 1).

Examples used in practice include the Gödel t-norm $\top(x, y) = min(x, y)$, the product t-norm $\top(x, y) = x \cdot y$ and the Lukasiewicz t-norm $\top(x, y) = max(x + y—1, 0)$. The notion of a t-cornorms is dual to that of a t-norm and is used to generalise logical disjunction. Formally, a tconorm is a binary operation $\perp$ on [0, 1] satisfying the aforementioned communatitivy, associativity and monotonicity properties and where the neutral element is 0 instead of 1 (that is, $\perp (x, 0) = x$). Examples of t-conorms used in practice are the Gödel t-conorm $\perp (x, y) = max(x, y)$, the product t-conorm $\perp(x, y) = x + y - x \cdot y$ and the Lukasiewicz t-conorm $\perp(x, y) = min(x + y, 1)$.

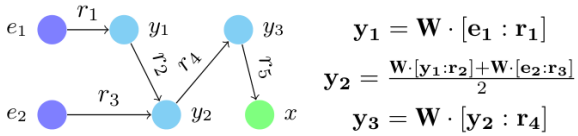


图 2: An example about how each intermediate variable (y_1, y_2 and y_3) node embedding in a complex logic query is calculated. Note that source nodes e1 and e2 and all relation nodes (r_1, r_2, r_3, r_4 and r_5) could be obtained di- rectly from the trained neural link prediction model.

3 Method

In this section, we present our approach Var2Vec to query answering over incomplete KGs. Our model relies on the availability of a trained neural link predictor $\mathcal{M}$ for a fi- nite vocabulary of entities $\mathcal{E}$and relations $\mathcal{R}$; it is compati- ble with most transductive link prediction methods proposed in the literature, such as TransE (Bordes et al. 2013),

DistMult (Yang et al. 2015), ComplEx (Trouillon et al. 2016), RotaE (Sun et al. 2018), and QuatE (Zhang et al. 2019). Indeed, the only requirement is that $\mathcal{M}$ can provide vector embeddings for each entity and relation in the vocabulary as well as a probability for each candidate fact. Then, given any admissible query $Q(x)$ and any KG $\mathcal{K}$ mentioning only relations from $\mathcal{R}$ and entities of $\mathcal{E}$, our model estimates the set of answers to $Q(x)$ with respect to the completion $\mathcal{K}^*$ of facts predicted to hold by $\mathcal{M}$.

At training time, we first train $\mathcal{M}$ in the usual way (or take a pre-trained link predictor instead); then, using the embeddings provided by $\mathcal{M}$, we train a weight matrix $\mathbf{W}$ that will be used to generate vector embeddings for the existentially quantified variables of any admissible input query at inference time.(2) The training of $\mathbf{W}$ is described in Section.

At inference time, we first traverse the input query $Q(x)$ to generate, using $\mathbf{W}$ and $\mathcal{M}$, vector embeddings for each existentially quantified variable; this is described in Section . Subsequently, as described in Section , we score each entity e in $\mathcal{E}$ to provide a likelihood of $Q(e)$ being true in $\mathcal{K}^*$; this allows us to rank the possible answers to the query.

3.1 Learning the Weight Matrix

Although a trained neural link predictor $\mathcal{M}$ can calculate a score for any fact $r(e_1, e_2)$ over its vocabulary, it cannot provide a score for atoms involving variables, such as $r(e_1, y)$ or $r(y_1, y_2)$, unless we first embed these variables as vectors. To this end, we propose to train and exploit a weight ma- trix $\mathbf{W} \in \mathbb{R}^{(k+k')\times k}$, where k and k' are the dimensions of the entity embeddings and the relation embeddings in $\mathcal{M}$, respectively. For instance, given an atom $r(e, y)$, the embed- ding of variable y is y = $\mathbf{W}\cdot[enc_{\mathcal{M}}(e) : enc_{\mathcal{M}}(r)]$, where [:] is the vector concatenation operation.

Given a set of training facts of the form $r(e_1, e_2)$, with $e_1, e_2 \in \mathcal{E} and r \in \mathcal{R}$, we minimise a L2-norm loss given next, where the elements of $\mathbf{W}$ are the only parameters:

$$\mathcal{L} = \| (\mathbf{W} \cdot [\text{enc}_{\mathcal{M}}(e_1) : \text{enc}_{\mathcal{M}}(r)]) - \text{enc}_{\mathcal{M}}(e_2)) \|_2. \tag{2}$$

Intuitively, $\mathbf{W}$ is trained to transform the embeddings of the first argument of a training fact and the fact′s relation into the embedding of the fact′s second argument. Note that, to train $\mathbf{W}$, we only require a set of training facts, which could be the same facts used to train $\mathcal{M}$; this is in contrast to query embedding models such as GQE (Hamilton et al. 2018), Query2Box (Ren, Hu, and Leskovec 2020), BetaE (Ren and Leskovec 2020) and ConE (Zhang et al. 2021), which usually require large amounts of training queries and answers in order to achieve good performance.

3.2 Generating Variable Embeddings

Given an admissible query $Q(x)$, a trained neural link pre- dictor $\mathcal{M}$, and a trained weight matrix W, we can generate vector embeddings for all existentially quantified variables in $Q(x)$. To this end, we traverse the query as described next where G_Q is the dependency graph of $Q(x)$. First, we mark the source terms t of G_Q as visited; these are entities be- cause the query is admissible, and hence we can obtain their embedding $\mathbf{t}$ = $\text{enc}_{\mathcal{M}}$(t); we also obtain the embedding of all relations r in the query as $\mathbf{r}$ = $\text{enc}_{\mathcal{M}}(r)$. Then, we repeat the following process until all existentially quantified variables in $Q(x)$ are marked as visited.

• Iterate through each unvisited existentially quantified variable y such that each atom in $Q(x)$ involving y is of the form $r_i(t_i, y)$ where each such $t_1, ..., t_n$ is an entity or an existentially quantified variable marked as visited.

—compute the embedding $\mathbf{y}$ of y as

$$mathbfy = \frac{\sum_{i=1}^{n} \mathbf{W} \cdot [\mathbf{t_i} : \mathbf{r_i}]}{n} \tag{3}$$

where t_i and r_i are the embeddings of term t_i (which is available at this stage due to admissibility of the query) and the relation r_i, respectively.

—Mark y as visited.

Example 0.1. *Consider the admissible query*

$$Q(x) = \exists y_1 \exists y_2 \exists y_3 \cdot (r_1(e_1, y_1) \wedge r_3(e_2, y_2) \wedge r_2(y_1, y_2) \wedge r_4(y_2, y_3) \wedge r_5(y_3, x)). \tag{4}$$

It′s dependency graph is provided in Figure 2. We

(2)Here and in the rest of the paper we omit bias terms for clarity

start by computing the embeddings e_1 and e_2 of entities e_1 and e_2 and the embeddings $r_1, ...r_5$ of relations $r_1, ..., r_5$ using the link predictor $\mathcal{M}$. We then process variable y_1 and obtain its embedding y_1 as $\mathbf{W} \bullet [e_1 : r_1]$. The next variable to process is y_2 and we obtain its embedding y2 as the average of $\mathbf{W} \bullet [e_2 : r_3]$ and $\mathbf{W} \bullet [y_1 : r_2]$. Finally, we can now obtain the embedding y_3 of variable y_3 as $\mathbf{W} \bullet [y_2 : r_4]$.

The process is clearly linear in the size of the query as each atom is considered only once. Note also that, to obtain variable embeddings at this stage, we do not define a special treatment of negation or disjunction as the query is traversed according to its dependency graph only.

3.3 Scoring for Query-Entity Pairs

Once we have obtained vector embeddings for all entities and existentially quantified variables mentioned in the input query $Q(x)$, we can exploit the decoding function $\text{dec}_{\mathcal{M}}$ to compute a score for each entity $e \in \mathcal{E}$ estimating the likelihood of $Q(e)$ being true in the completion $\mathcal{K}^*$. For each entity e, the computation of the score is performed inductively on the structure of $Q(e)$. Formally, let us fix a t-norm function $\top$ and a t-conorm function $\bot$. Then, the scoring function σ maps each admissible FOL sentence φ constructed according to the grammar in Equation (1) over the relevant vocabulary to a real-valued score $\sigma(\varphi) \in [0, 1]$ as given next, where for a term t (constant or existentially quantified variable) and relation r occurring in φ, we use t and r to respectively denote their vector embedding computed as described in Section.

- if φ is an atom of the form $r(t_1, t_2)$, for t_1 and t_2 terms, then $\sigma(\varphi) = \text{dec}_M(t_1, r, t_2)$;
- if φ is a negative literal of the form $\neg r(t_1, t_2)$, then $\sigma(\varphi) = 1 - \sigma(r(t_1, t_2))$;
- if $\varphi = \varphi_1 \wedge \varphi_2$, then $\sigma(\varphi) = \top(\sigma(\varphi_1), \sigma(\varphi_2))$;
- if $\varphi = \varphi_1 \vee \varphi_2$, then $\sigma(\varphi) = \bot(\sigma(\varphi_1), \sigma(\varphi_2))$; and
- if φ is of the form $\exists t.\varphi_1$,then $\sigma(\varphi_1)$.

Note that the computation of the score for a given entity is linear in the size of the query; furthermore, answering the query requires a score computation for each entity in the KG. Once, all scores have been computed, the possible answers can be ranked before they are returned to the user.

4 Experiments

We have implemented our Var2Vec model and evaluated its performance against state-of-the-art baselines.

In this section, we describe the results of our evaluation on a suite of an established benchmark for query answering over incomplete KGs.

Datasets and Queries We consider the query answering benchmark proposed in (Ren, Hu, and Leskovec 2020). The benchmark provides three standard KGs commonly used in the KG completion literature: FB15k (Bordes et al.2013), FB15k-237 (Toutanova and Chen 2015), and a subset of the NELL995 KG (Xiong, Hoang, and Wang 2017). The benchmark splits the facts in each KG into training, validation and testing subsets, where the training subset is contained in the validation subset and the validation subset is in turn contained in the testing subset. Given a query Q and a benchmark KG $\mathcal{K}$, let us denote with $[Q]^{train}_{\mathcal{K}}$, $[Q]^{val}_{\mathcal{K}}$ and $[Q]^{test}_{\mathcal{K}}$ (where) the set of answers to Q with respect to the training, validation, and testing subsets of K, respectively. We evaluate on

KG	Training		Validation		Test	
	1p	others	1p	others	1p	others
FB15k	273,710	273,710	59,097	8,000	67,016	8,000
FB15k-237	149,689	149,689	20,101	5,000	22,812	5,000
NELL995	107,982	107,982	16,927	4,000	17,034	4,000

表 1: Number of training, validation, and test queries gen- erated for different query patterns.

the simplest pattern and it corresponds to facts, whereas the remaining patterns represent increasingly complex query structures. The queries used for training, validation, and testing are obtained by instantiating each query pattern multiple times for each benchmark KG; this is done by choosing suitable relations and entities from the KG. Table 1 summarises the number of instantiations of each query pattern and each KG provided by the benchmark.

Method	Supported Operators
GQE (Hamilton et al. 2018)	∃, ∧
Query2Box (Ren, Hu, and Leskovec 2020)	∃, ∧, ∨
BetaE (Ren and Leskovec 2020)	∃, ∧, ∨, ¬
MLP (Amayuelas et al. 2021)	∃, ∧, ∨, ¬
ConE (Zhang et al. 2021)	∃, ∧, ∨, ¬
CQD (Arakelyan et al. 2021)	∃, ∧, ∨
FuzzQE (Chen, Hu, and Sun 2022)	∃, ∧, ∨, ¬
Var2Vec	∃, ∧, ∨, ¬

表 2: Considered baselines and our approach Var2Vec.

Baselines We compare Var2Vec against the state-of-the- art baselines indicated in Table 2. To perform the experiments, we used the code for the relevant baseline provided by the authors; the only exception was FuzzQZ (Chen, Hu, and Sun 2022), where we reported the results provided in their paper since the code for the system is not publicly available. All baselines support only admissible queries as described in the preliminaries; however, GQE, Query2Box and CQD are restricted to admissible queries without negation. We did not include the beam search scheme of CQD scheme as a baseline in our experiments since, as show in Figure 4, it is computationally too expensive; for example, a single 12GB Titan X GPU returned a memory error for a 5000-batch query inference on NELL995; as a result, we used only the continuous optimisation scheme of CQD. It is worth reiterating that both our approach and CQD require only facts (i.e.,queries of pattern 1p) for training, whereas training for the remaining baselines also requires the complex queries and their answers in the benchmark.

Model Details For each dataset, we pretained a link predictor (ComplexE) only using the 1p queries from their training dataset. The number of training epochs was 100 and the embedding size of both the entity and the relation was 2000. The dimension of $\mathbf{W}$ was 4000×2000. We used the Adam Optimizer (Kingma and Ba 2014) with learning rates lr=0.1/0.01/0.001 to optimize $\mathbf{W}$ and used the best one on the validation set. The batch size was 1000. In our experiments, by default, we use product T-norm ($\perp(x, y) = x \cdot y$) and product T-conorm ($\top(x, y) = x + y - x \cdot y$).

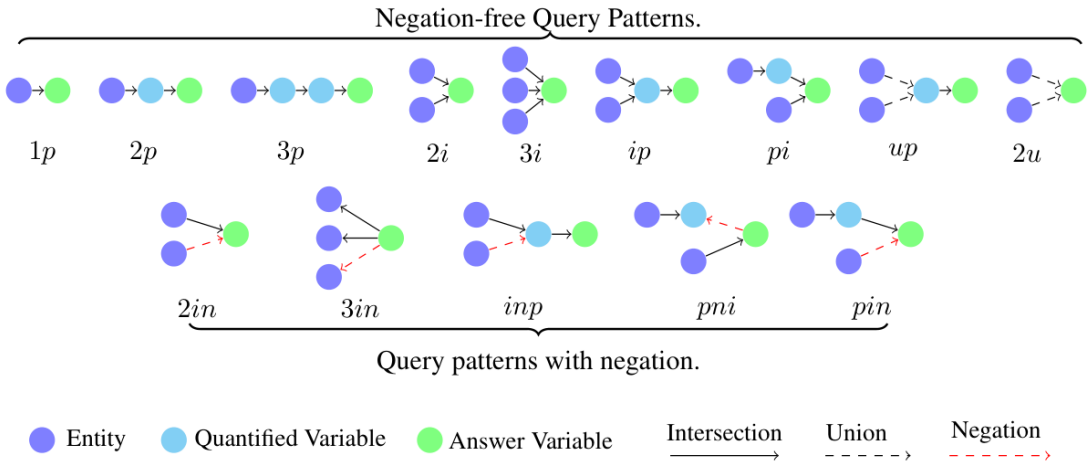


图 3: Query patterns considered in the experiments. In the naming of query structures, p, i, u and n stand for Projection, Intersection, Union and Negation, respectively.

Evaluation Metrics Following prior work (Ren, Hu, and Leskovec 2020; Ren and Leskovec 2020), we adopted the Mean Reciprocal Rank (MRR) metric for our evaluation, which is calculated as given next for a test query Q and a test KG $\mathcal{K}$, where $rank_e$ denotes the rank of entity e as a query answer amongst all entities in the KG:

Main Results. Table 3 summarises the results of our evaluation on all three benchmark KGs, where Avg_e (respectively, Avg_n) represents the average MRR for all queries in the benchmark corresponding to negation-free patterns (respectively, queries corresponding to patterns with negation). Overall, the performance of our approach is highly com-

petitive. Compared with GQE, Query2Box and BetaE, our method achieves an average improvement on MRR of 37.7%, 22.9% and 46.2% respectively for queries without negation, and also outperforms significantly all of these approaches for queries with negation. Our approach also outperforms ConE and MLP both for queries with and without negation, especially for complex query patterns such as ip and pi, which are patterns unseen during training for all models; this suggests that our approach generalises better to new query patterns than the baselines. Compared to CQD, our approach also displays better average performance for all three benchmark KGs. Finally, we compared our approach to FuzzQE, which also exploits the availability of a neural link predictor; we observe that our approach also outperforms FuzzQE in most cases, with the only exception of queries with negation on NELL995.

Figure 4 provides training and average inference times for all the evaluated approaches on FB15k. We can see that our approach can be trained and applied very efficiently. In particular, our approach is 12/46 times faster than CQD- CO/CQD-Beam during inference and significantly faster than embedding-based models during training.

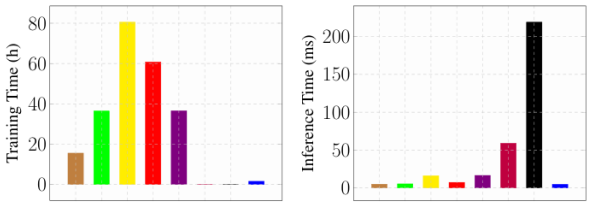


图 4: Training and inference times on FB15k. Inference time refers to the average time per query. From left (brown) to right (blue), it represents GQE, Query2Box, BetaE, MLP, ConE, CQD-CO, CQD-Beam and Var2Vec, respectively.

Method	Avge	Avgn	1p	2p	3p	2i	3i	ip	pi	2u	up	2in	3in	inp	pin	pni
NELL995																
GQE	18.6	-	32.8	11.9	9.6	27.5	35.2	14.4	18.4	8.5	8.8	—	—	—	—	—
Q2B	22.9	-	42.2	14.0	11.2	33.3	44.5	16.8	22.4	11.3	10.3	—	—	—	—	—
CQD-CO	28.8	-	60.8	18.3	13.2	41.0	41.5	22.5	30.3	17.6	13.7	—	—	—	—	—
BetaE	24.6	5.9	53.0	13.0	11.4	37.6	47.5	14.3	24.1	12.2	8.5	5.1	7.8	10.0	3.1	3.5
ConE	27.2	6.4	53.1	16.1	13.9	40.0	50.8	17.5	26.3	15.3	11.3	5.7	8.1	10.8	3.5	3.9
MLP	25.0	6.0	52.7	15.4	14.0	36.4	45.4	15.8	22.1	13.2	10.0	5.1	8.0	10.0	3.6	3.6
FuzzQE	27.1	7.3	57.6	17.2	13.3	38.2	41.5	27.0	19.4	16.9	12.7	9.1	8.3	8.9	4.4	5.6
Var2Vec	30.1	7.5	60.8	18.0	11.9	42.2	52.2	22.7	30.9	19.2	12.2	6.8	8.8	10.6	5.4	5.9
FB15k-237																
GQE	16.3	—	35.0	7.2	5.3	23.3	34.6	10.7	16.5	8.2	5.7	—	—	—	—	—
Q2B	20.1	—	40.6	9.4	6.8	29.5	42.3	12.6	21.2	11.3	7.6	—	—	—	—	—
CQD-CO	21.8	—	46.7	9.5	6.3	31.2	40.6	23.6	16.0	14.5	8.2	—	—	—	—	—
BetaE	20.9	5.4	39.0	10.9	10.0	28.8	42.5	12.6	22.4	12.4	9.7	5.1	7.9	7.4	3.6	3.4
ConE	23.2	5.9	42.3	12.7	10.7	32.6	46.9	13.6	25.2	14.2	10.6	5.4	8.6	7.8	4.0	3.6
MLP	22.1	6.7	41.4	11.7	9.9	32.1	44.6	13.0	24.5	12.6	9.3	6.4	10.6	8.0	4.6	4.4
FuzzQE	21.8	6.6	44.0	10.8	8.6	32.3	41.4	22.7	15.1	13.5	8.7	7.7	9.5	7.0	4.1	4.7
Var2Vec	23.4	6.4	46.7	10.8	8.4	33.5	46.5	16.3	22.5	16.6	8.8	5.4	10.5	6.2	4.4	5.3
FB15k																
GQE	28.0	—	54.6	15.3	10.8	39.7	51.4	19.1	27.6	22.1	11.6	—	—	—	—	—
Q2B	38.0	—	68.0	21.0	14.2	55.1	66.5	26.1	39.4	35.1	16.7	—	—	—	—	—
CQD-CO	48.6	—	89.4	27.5	15.0	76.9	80.8	35.1	46.4	42.7	23.5	—	—	—	—	—
BetaE	41.6	11.8	65.1	25.7	24.7	55.8	66.5	28.1	43.9	40.1	25.2	14.3	14.7	11.5	6.5	12.4
ConE	49.7	14.8	73.2	33.7	29.0	64.6	73.6	35.5	50.9	55.3	31.4	17.9	18.7	12.5	9.8	15.1
MLP	41.6	13.9	66.9	29.3	24.4	56.1	66.2	26.7	44.8	33.5	26.4	16.0	17.3	13.2	8.3	14.6
FuzzyQE*	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Var2Vec	51.0	18.9	89.4	26.0	16.6	79.0	82.6	34.2	46.0	62.8	22.1	23.6	31.4	9.8	9.8	19.7

表 3: MRR results (%) of baselines and our model on benchmark queries grouped by query pattern. Avge and Avgn denote the average MRR on queries with and without negation, respectively.

Ablation Studies. We first analysed the influence of the chosen link predictor on our results. Our system uses ComplexEx by default. If we choose DisMult instead, we can observe in Table 4 a slight decrease in performance. This is unsurprising given that ComplexEx ourperforms DistMult in link prediction tasks over the benchmark KGs.

We also analysed the influence of the choice of t-norm and t-conorms, which play an important role in computing the scores for query answers. Our system uses product t- norms and t-conorms by default, and we also experimented with G¨odel t-norms and t-conorms. Table 5 shows that using product t-norms and t-conorms leads to superior performance on all benchmark KGs.

Finally, we explored the possibility of replacing the weight matrix **W** with a fully-connected neural network equipped with ReLU activations. We considered architectures with one and two hidden layers in which ReLU is applied following matrix application in each hidden layer. The results we obtained on the FB15k KG are summarised in Table 6. We can observe that, as the number of layers increases, the training times increase significantly but the performance of the model does not show a noticeable improvement. These results thus speak in favour of using a simple weight matrix instead of a more complex neural architecture for computing variable embeddings.

Case Study. In this section we discuss the variable embeddings obtained by our approach. We chose as a case study the test query in the NELL995 benchmark listing ” all countries where an official language of the country is used for teaching

Method	Avg	1p	2p	3p	2i	3i	ip	pi	2u	up
NELL995										
DisMult	28.1	58.6	13.9	10.3	40.7	51.0	18.9	28.9	18.7	11.5
ComplexE	30.1	60.8	18.0	11.9	42.2	52.2	22.7	30.9	19.2	12.8
FB15k-237										
DisMult	21.6	45.4	8.0	6.6	31.5	44.0	13.2	21.0	16.2	8.1
ComplexE	23.2	46.7	10.3	7.1	33.5	46.5	16.3	22.5	16.6	8.8
FB15k										
DisMult	41.8	73.4	17.6	15.4	60.5	68.8	30.1	39.5	50.5	20.5
ComplexE	51.0	89.4	26.0	16.6	79.0	82.6	34.2	46.0	62.8	22.1

表 4: MRR results (%) with DisMult and ComplexE.

Method	Avg	1p	2p	3p	2i	3i	ip	pi	2u	up
NELL995										
MM	25.9	60.8	18.0	7.1	39.7	46.4	15.8	13.8	19.9	11.3
PP	30.1	60.8	18.0	11.9	42.2	52.2	22.7	30.9	19.2	12.8
FB15k-237										
MM	19.4	46.7	8.8	6.9	30.9	39.6	11.5	6.7	17.2	6.5
PP	23.2	46.7	10.3	7.1	33.5	46.5	16.3	22.5	16.6	8.8
FB15k										
MM	45	89.4	16.8	11.4	80.5	84.1	21.8	10.5	74.1	16.4
PP	51.0	89.4	26.0	16.6	79.0	82.6	34.2	46.0	62.8	22.1

表 5: MRR results (%) with different T-norms. MM represents the Min-Max G¨odel combination and PP for the Product-Product combination.

Divinity at universities” . This is a conjunctive query written as follows, with $Divinity$ an entity NELL995:

$$Q(x) = \exists y_1 \exists y_2 \cdot (\text{ CourseOf (Divinity, } y_1) \wedge TeachLanguage\,(y_1, y_2) \wedge OfficialLanguage\ (y_2, x)) \tag{5}$$

An answer to this query can be obtained by matching the answer variable x to US and the existentially quantified variables y_1 and y_2 to $HarvardUniversity$ and $English$, respectively. Intuitively, we would expect the embedding of variables y_1 and y_2 obtained as described in Section to be close to the embeddings for university entities and language entities, respectively.

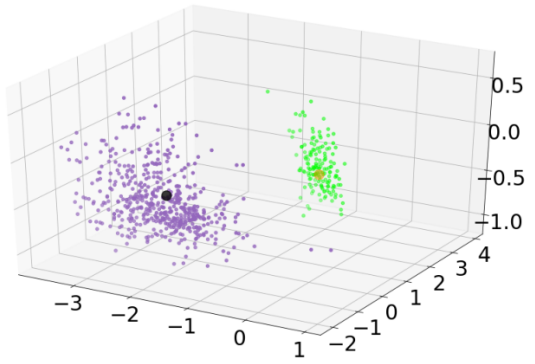


图 5: Visualisation of entity and variable embeddings for the case study. Th black and yellow points represents the embedding of variable y_1 and y_2, respectively; and the purple and green clusters denote the embeddings of university entities and language entities, respectively.

To verify this intuition, we have performed a Principal Component Analysis (PCA) (Abdi and

Williams 2010) on the embeddings. The results are depicted in Figure 5. Going from left to right, the first cluster of points represents the embeddings of University entities (in purple) and the embedding of variable y_1 (in black), whereas the second cluster of points represent embeddings of language entities (green points) and the embedding of y_2 (yellow point). We can observe that variable embeddings show strong correspondence with the embedding of their representative entities in terms of their relative positions in Figure 5. This illustrates that the learnt weight matrix **W** is able to adequately embed variables in queries with joins.

5 Related Work

Knowledge graph completion is the problem of completing a KG with missing facts that are likely to hold in the domain of interest. Neural link predictors are models capable of scoring the likelihood of a particular fact and/or ranking the most likely answers to an atomic query of the form $Q(x) = r(e, x) or Q(x) = r(x, e)$ for r a relation and e an entity in the KG. Knowledge graph completion has received a great deal of attention in recent years and a wide range of neural link predictors have been proposed. Prominent examples include TransE (Bordes et al.2013), DistMult (Yang et al.2015), ComplEx (Trouillon et al.2016), and RotaE (Sun et al. 2018). We refer the readers to (Rossi et al.2021) for a more detailed discussion about these models. It is worth mentioning, however, that the neural link predictors that are compatible with our approach are $transductive$—that is, they score individual facts by first learning embeddings for entities and relations in the KG and as a result they can only make predictions for entities seen during training. In recent years, however, there has also been increasing interest on $inductive$ link predictors, which are able to make predictions for KGs involving arbitrary entities (Hao et al. 2020; Teru, Denis, and Hamilton 2020; Liu et al. 2021).

In this paper, we focus on the query answering problem over incomplete KGs, which generalises link prediction to FOL queries that go beyond simple facts. This problem is receiving increasing attention, and prominent approaches include GQE (Hamilton et al.2018), Query2box (Ren, Hu, and Leskovec 2020), BetaE (Ren and Leskovec 2020), MLP (Amayuelas et al.2021) and ConE (Zhang et al.2021). Specifically, GQE is one of the earliest approaches supporting only admissible conjunctive queries. Query2Box further supports disjunction by representing queries as a box in a latent space; box embeddings, however, cannot support negation since the complement of a box in the Euclidean space is no longer a box. Beta embeddings (Ren and Leskovec 2020) were later proposed so address this limitation and support admissible queries involving all four Boolean operations. Subsequent approaches, such as ConE and MLP, improved model performance by introducing increasingly complex architectures; these query embeddingbased approaches are, however, data hungry and usually require millions of training queries to achieve a satisfying performance. CQD (Arakelyan et al.2021) exploits trained neural link predictors together with fuzzy logic operators to avoid the need for complex queries during training. CQD requires only facts for training and displays improved out-of-distribution generalisation. CQD, however, does not support negation and has low inference efficiency.

6 Conclusion

We have presented a novel approach to query answering over incomplete KGs that addresses some of the key limitations in prior work. First, we support admissible queries involving all Boolean operators. Second, we require only facts for training, and training can be performed very efficiently. Finally, our approach displays superior accuracy while at the same time significantly reducing inference times.

7 Acknowledgments

This work was supported in whole or in part by the EPSRC projects OASIS (EP/S032347/1), ConCuR (EP/V050869/1) and UK FIRES (EP/S019111/1), the SIRIUS Centre for Scalable Data Access, and Samsung Research UK. For the purpose of Open Access, the authors have applied a CC BY public copyright

licence to any Author Accepted Manuscript (AAM) version arising from this submission. The authors would like to thank Shuai Zhang, Qi Liu, and Yue Zhang for their insightful discussions and support.

参考文献

[1] Abdi,H.; and Williams,L. J. 2010. Principal component analysis. Wiley interdisciplinary reviews: computational statistics, 2(4): 433—459.

[2] Amayuelas, A.; Zhang, S.; Rao, X. S.; and Zhang, C. 2021. Neural Methods for Logical Reasoning over Knowledge Graphs.In International Conference on Learning Representations (ICLR).

[3] Arakelyan, E.; Daza, D.; Minervini, P.; and Cochez, M. 2021. Complex Query Answering with Neural Link Predictors. In International Conference on Learning Representations (ICLR).

[4] Bordes, A.; Usunier, N.; Garcia-Duran, A.; Weston, J.; and Yakhnenko, O. 2013. Translating embeddings for modeling multi-relational data. Advances in Neural Information Processing Systems (NeurIPS).

[5] Chen, X.; Hu, Z.; and Sun, Y. 2022. Fuzzy Logic Based Logical Query Answering on Knowledge Graphs. In Proceedings of the AAAI Conference on Artificial Intelligence,volume 36, 3939—3948.

[6] Gupta, M. M.; and Qi, J. 1991. Theory of T-norms and fuzzy inference methods. Fuzzy sets and systems,40(3): 431—450. Hájek, P. 2013. Metamathematics of fuzzy logic, volume 4. Springer Science & Business Media.

[7] Hamilton, W. L.; Bajaj, P.; Zitnik, M.; Jurafsky, D.; and Leskovec, J. 2018. Embedding Logical Queries on Knowledge Graphs. In Advances in Neural Information Processing Systems (NeurIPS), 2030—2041.

[8] Hao, Y.; Cao, X.; Fang, Y.; Xie, X.; and Wang, S. 2020. Inductive Link Prediction for Nodes Having Only Attribute Information. In International Joint Conference on Artificial Intelligence (IJCAI), 1209—1215.

[9] Heist, N.; Hertling, S.; Ringler, D.; and Paulheim, H. 2020. Knowledge Graphs on the Web-An Overview. Knowledge Graphs for eXplainable Artificial Intelligence, 3—22.

[10] Hogan, A.; Blomqvist, E.; Cochez, M.; d'Amato, C.; Melo, G. d.; Gutierrez, C.; Kirrane, S.; Gayo, J. E. L.; Navigli, R.; Neumaier, S.; et al. 2021. Knowledge graphs. Synthesis Lectures on Data, Semantics, and Knowledge, 12(2): 1—257.

[11] Kingma, D. P.; and Ba, J. 2014. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

[12] Li, F.-L.; Chen, H.; Xu, G.; Qiu, T.; Ji, F.; Zhang, J.; and Chen, H. 2020. AliMeKG: Domain knowledge graph construction and application in e-commerce. In ACM International Conference on Information and Knowledge Management (CKIM), 2581—2588

[13] Liu, S.; Grau, B. C.; Horrocks, I.; and Kostylev, E. V. 2021. INDIGO: GNN-Based Inductive Knowledge Graph Completion Using Pair-Wise Encoding. In Advances in Neural Information Processing Systems (NeurIPS).

[14] Noy, N.; Gao, Y.; Jain, A.; Narayanan, A.; Patterson, A.; and Taylor, J. 2019. Industry-scale Knowledge Graphs: Lessons and Challenges: Five diverse technology companies show how it's done. Queue, 17(2): 48—75.

[15] Ren, H.; Hu, W.; and Leskovec, J. 2020. Query2box: Reasoning over Knowledge Graphs in Vector Space Using Box Embeddings. In International Conference on Learning Representations (ICLR).

[16] Ren, H.; and Leskovec, J. 2020. Beta embeddings for multi-hop logical reasoning in knowledge graphs. Advances in Neural Information Processing Systems (NeurIPS), 33:19716—19726.

[17] Rossi, A.; Barbosa, D.; Firmani, D.; Matinata, A.; and Merialdo, P. 2021. Knowledge graph embedding for link prediction: A comparative analysis. Acm Transactions on Knowledge Discovery from Data (TKDD), 15(2): 1—49.

[18] Sun, Z.; Deng, Z.-H.; Nie, J.-Y.; and Tang, J. 2018. Ro-tatE: Knowledge Graph Embedding by Relational Rotation in Complex Space. In International Conference on Learning Representations (ICLR).

[19] Teru, K.; Denis, E.; and Hamilton, W. 2020. Inductive relation prediction by subgraph reasoning. In International Conference on Machine Learning (ICML), 9448—9457.

[20] Toutanova, K.; and Chen, D. 2015. Observed versus latent features for knowledge base and text inference. In Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality, 57—66. Beijing, China: As- sociation for Computational Linguistics.

[21] Trouillon, T.; Welbl, J.; Riedel, S.; Gaussier, É.; and Bouchard, G. 2016. Complex embeddings for simple linkprediction. In International Conference on Machine Learning (ICML), 2071—2080. PMLR.

[22] Xiong, W.; Hoang, T.; and Wang, W. Y. 2017. DeepPath: A Reinforcement Learning Method for Knowledge Graph Reasoning. In Empirical Methods in Natural Language Processing (EMNLP), 564—573.

[23] Xu, D.; Ruan, C.; Korpeoglu, E.; Kumar, S.; and Achan, K. 2020. Product knowledge graph embedding for e-commerce. In International Conference on Web Search and Data Mining (WSDM), 672—680.

[24] Yang, B.; Yih, S. W.-t.; He, X.; Gao, J.; and Deng, L. 2015. Embedding Entities and Relations for Learning and Inference in Knowledge Bases. In International Conference on Learning Representations (ICLR).

[25] Zhang, S.; Tay, Y.; Yao, L.; and Liu, Q. 2019. Quaternion knowledge graph embeddings. Advances in Neural Information Processing Systems (NeurIPS), 32.

[26] Zhang, Z.; Wang, J.; Chen, J.; Ji, S.; and Wu, F. 2021. Cone: Cone embeddings for multihop reasoning over knowledge graphs. Advances in Neural Information Processing Systems (NeurIPS).

作者信息

Dingmin Wang Department of Computer Science, University of Oxford, UK

Yeyuan Chen The School of Computer Science and Technology, Xi'an Jiaotong University, China

Bernardo Cuenca Grau Department of Computer Science, University of Oxford, UK

All authors make equal contributions to this paper.

砥砺——社科

从言与意的张力谈对《庄子》的阅读体认——以庄惠“濠梁之辩”寓言为例

焦成城

摘要

言意关系是庄子哲学的重要议题。在庄子哲学中，“言”与“意”有着多重含义，集中体现在语言哲学、心灵哲学以及形而上学本体论等方面。以往某些研究认为庄子完全反对一切言语，不仅要无言、去言甚至要无言、绝言，因而陷入了不可知论。本文通过对《庄子》相关文本进行考察，认为庄子从“言不尽意”这一现实境遇出发，经由“得意忘言”的实现路径，最终朝向超越“言意”的达道境界。这其实是在强调语言的局限性，属于可知论。并且，从言意张力这一崭新视角切入“濠梁之辩”的文本，我们发现：与惠子经验层面的“以物观之”、最终将“道”具体化因而片面化不同，庄子对真意即“道”的把握是通过两方面实现的：（1）非逻辑的直觉体悟；（2）“天人合一”的超验体验（“以道观之”）。庄子所追求的是语言“即言即道”的自由本真之逍遥境界。

关键词：庄子；言意之辨；言不尽意；得意忘言；达道；“濠梁之辩”.

1 引言：何为“言”与“意”

魏晋南北朝时期“言意之辩”可谓中国哲学史上的一大学术论战，其最早可追溯到《周易》，如“书不尽言，言不尽意”（《周易·系辞上》）[1]，所涉及的是卦象、卦言（卦辞和爻辞）和卦义三者之间的张力关系。甲骨文的“言”字为口上加辛（薪），表示口在说话的嘈杂声 [2]，后来用作名词，又引申为言语与文字（文字为固化的语言），即声波或笔画等物理实在（庄子的“三言”与“庄语”后文展开）。在庄子哲学中，“意”的涵义较为复杂，综合多方论述 [3]，本文划分如下：（1）形而下器物之“意思”或“意义”：“尽意”与“明义”与“诠理”是相近的 [4]。（2）心理层面上的“意念”或“意象”：精神性的观念以及人头脑中的理智直观物，因此与心外之“言”相异，《说文》：“意，志也，从心音”。（3）形而上宇宙论的“真意”（这在很大程度上是超越于言意的）：支配宇宙万物的根本规律，即道家所谓之“道”，如《老子》所说：“道可道，非常道。名可名，非常名”，又如“吾不知其名，字之曰道，强为之名曰大”（《老子》二十五章）[5]，即是一种不可言说的形而上之道，是人类认识的终极。而庄子的言意观便继承于其道“无”本体论，“道不可言，言而非也”（《庄子·知北游》）[6]。因此，庄子的言意观涉及语言的逻辑形式与义理内涵（语言哲学）、心中意念与心外言语（心灵哲学）以及得道与言语（言道之辩）这三个层面。

结合上述分析，不难发现，二者存在着所谓“知者不言，言者不知”(《老子》五十六章、《庄子·知北游》)[7] 的二律背反或“言道悖论”，即道（意）不可言、但又不能不说。因为意一旦被言说就被规定为一个有限的存在者，而如果不加言说其丰富内涵无法窥见，从而成为一个自在的绝对之物。那么，庄子是如何处理“言”和“意”二者间张力的？其真的诚如一些批评者所指出的那样“忘言、去言乃至于绝言”吗？针对上述问题，本文立足《庄子》文本进行考察，其安排如下：第二节我将对庄子言意的相关论述进行梳理，并

指出庄子言意观“言不尽意-得意忘言-超越言意”的三重逻辑；第三节将聚焦于庄惠“濠梁之辩”寓言，来进一步例证这一方案对庄子言意关系的解释有效性，最后我将指出该思想的后世回响，并从中西比较的视角出发，做一些延伸思考。

2 庄子“言意观”的三重逻辑

2.1 出发点：言不尽意

庄子首先通过对沉湎经书进行批判，在语言哲学层面阐明：尽管语言形式具有表意的功能，但无法完全传达圣人之意的内容，因而突出“意”的首要性，如“书不过语，语有贵也。语之所贵者，意也。意之所随者，不可以言传也”（《庄子·天道第十三》）[8]，因而世人只得“贵其非贵”。接着，其用“轮扁祈轮”的寓言进一步说明“得之于手而应之于心”的精妙之道“口不能言”，即有些“意”无法“言”。言有尽而意无穷，真正的“意”是不可传授的。这是因为一方面圣人已死，即在“意”之主体已然非存在的情况下，留下的言显然无法完全通达圣人本意；同时，从历时性思考，“言”在时间中绵延，面向未来所敞开之“意”也会因所处境遇不同而有着多种可能朝向。此乃言意观的第一层，即“言不尽意”。

2.2 实现路径：得意忘言

“荃者所以在鱼，得鱼而忘荃；蹄者所以在兔，得兔而忘蹄。言者所以在意，得意而忘言。吾安得夫忘言之人而与之言哉?”（《庄子·外物第二十六》）[9]。同获兔可丢蹄（套索）和得鱼可弃荃（竹篓）一样，只要把握住了“意”，“言”不过只是通达于“意”的方式、工具和手段。借用韦伯的话说，“言”处于工具理性的地位，而作为价值理性之“意”的表达才是庄子所看重的，这同样也是庄惠二人多次辩论的立场分歧。此处得意忘言之人即得道之人，正如成玄英所说“鱼兔得而忘荃蹄，玄理明而名言绝”[10]。所忘的是“小言”“俗言”，值得追求的应是“大言”“至言”即天道。反过来说，若只拘泥于“言”的现象层面则不利于把握事物“意”之本质，便无法通达天道。因此“言由意生”，“意”是根本，应当“得意忘言”。

2.3 最终朝向：超越“言意”

可以言论者，物之粗也；可以意致者，物之精也。言之所不能论，意之所不能察致者，不期精粗焉”（《庄子·秋水第十七》）[11]。言论即一种知性概念的表达，而意致则倾向于一种理智直观。“精粗”所描述的是物，是形而下的论域，而道是“物物而不物于物”者。借用西方经验论者话说，“不期精粗”是物本身内在所固有的第一性质，而“物之精粗”是事物与主体发生关联的第二性质。因此，可以言论和意致的世界，也即可用人之感性知性能力所把握的，其实仍处于“名言之域”，即合于人道的方内；而不可以言论意至的“不期精粗”的“超名言之域”，才是合乎天道的方外之境。

如此，两个领域似乎界限分明，那么如何实现从“言意”的经验层面上升的超验之大“道”呢？在庄子看来，从“意”的角度，要以“虚静”“无待”来体道悟道，即所谓“心斋”“坐忘”的方法论或工夫论来“去知”“去己”，从而实现“与道合一”；而从“言”的角度，则要采用非同寻常的方式“言不可言（者）”。庄子指出当时言语的困境为“以天下为沈浊，不可与庄语。以卮言为曼衍，以重言为真，以寓言为广。”（《庄子·天下》）[12] 钟泰先生注曰：“‘庄语’，正言也”/[12]。只有用并非庄重不苟、正襟危坐的言论“寄言出意”，才可曲径通幽、言不可言。“三言”即寓言（借象征性的语言来引发联想或类比）、重言（通过肯定否定、揶揄调侃固化名人之言）、卮言（“语义滑转不定、随说随扫以破除成心的语言”）[13][14]。实际上，从现代语言哲学或修辞学来讲，“三言”的本质是隐喻，通过表面说 A1 来引发 A2, A3…An 来揭示本体和喻体的关系，而这正是中国文化独有的求道（真理）的方式。

3 “言意”张力下的庄惠“濠梁之辩”

在先秦哲学中，注重讨论言意关系以及名实关系则首提名家，其中，以公孙龙子与惠施最具代表性。庄惠之交与之辩历来是庄子哲学中研究的一大重点，因为“惠施是内七篇中提到的唯一和庄子对话的人物”[15]。庄惠之辩主要出现于《逍遥游》《德充符》与《秋水》三篇，并且诸篇皆以惠施之问答收尾。其中，《逍遥游》中的“大瓠之種”“大树樗”所涉及的主要是无用与有用之辩，而《德充符》中同样主要讨论天情与人情、德行与性命，两则辩论与言意主题联系并不紧密。“濠梁之辩”历来被学者所重视和讨论，但诸家在一定程度上忽视了言意观的视角。因此，下文我将结合第二节中庄子言意观的三重逻辑，来从言意张力的视角切入该篇。

3.1 “濠梁之辩”何以体现“言不尽意”？

具体来说，“濠梁之辩”的“言不尽意”之张力体现在以下方面：其一，若将整个篇章视为一个整体，其同样向读者传达了不同于表面可视的文字之言

（对话的逻辑结构）的话外之话、弦外之音（寄托或隐喻的义理）。其二，双方在濠上试图通过对话或论辩等“言”的形式，试图使得此心之“意”跃出自身的局限，切入他者的彼心。一人之“言”何以“尽”他人之心，此即心灵哲学中的所谓“他心问题”。与庄子以齐物立论、追求人与人、乃至人与鱼等有生之灵的经验互通不同（即“主体间性”），惠子则秉承“言”所传达之“意”完全依赖于主体这一逻辑前提，因而“我非子，固不知子”，实质是坚持不同主体之心灵难以完全通达。两人预设立场的分歧也造就了言意不可通达。

3.2 惠子的“执拗于言”与庄子的“得意忘言”

语言是意义的符号，而意义由思想所构成。惠子秉承一种“察”“辩”分析的科学精神，对庄子有感物情而所叹鱼乐之“意”进行质疑，庄子起初跟随惠子的逻辑后来又言“请循其本”，回答“知鱼之乐”的来源是“知之濠上也”。这看似是转移话题或不合逻辑的机智婉语，实则是庄子的所谓“卮言”。因为他发现惠子即言说主体固执于“言”，而其一旦有所执念，丢不掉维特根斯坦意义上的“梯子”，陷于技术理性的窠臼，便无法上升到形而上境界。正如冯友兰所说，由每个人语言所表达的思想皆源于其自身“特殊的有限的观点所形成的意见”，而有限即意味着片面，意识不到这一点，便自以为是、以别人为非 [16]。双方立场不同、思维方式不同、针对的问题也相异，如此在“言”上争辩没有任何价值。正如庄子假借公子牟批判公孙龙所说：“子乃规规然而求之以察，索之以辩，是直用管窥天，用锥指地也，不亦小乎？”（《庄子·秋水》）[17] 语言本来由人产生，但又反过来重新规训人，这在一定程度上也造成了“异化”。而要扬弃异化，必须“得意忘言”。

3.3 惠子的形下“言意”之困境与庄子的超越“言意”之境界

在《天下篇》中，庄子认为“惠施多方，其书五车，其道舛驳，其言也不中。”（《庄子·天下》）[18]，其所提倡的“历物十意”主张以“合同异”来辨物，是一种类似近代科学的纯学术的认识论，不涉及方和术，与庄子所强调的“内圣外王”之道大相径庭 [19]。身为一国之相的惠子既不追求治国安邦的谋略，也不关注社会伦理道德问题，而只“专决于名”，也由此引发了上述的“言意”之困境。郭象也指出：“累于形名，以庄语为狂而不信，故不与也。” [20] 哲学的使命正在于说不可说，惠子的“以物观之”恰恰是庄子所推崇的天道之整全“以道观之”所批评的。因为“道”不是“物”，其是包含一切规定性的整全或大全，从经验层面试图对“道”具体化的同时也使之片面化，正如作为原初整全的混沌被凿七窍而死（《庄子·应帝王》）[21]。这也是上世纪西方逻辑实证主义试图找到一种万能的理想技术语言以及先秦孔子尝试“正名”失败的原因。正如海德格尔所说，“语言是存在的家”，而语言总是在说什么，即对象化的语言反而遮蔽了此在的真理（道），语言便丧失了本质 [22]。因此，与惠子限于形而下的实用层面不同，庄子则追求的是语言“即言即道”的自由本真之逍遥境界。

4 结论

综上所述，本文得出以下三点结论：第一，言意关系是庄子哲学的重要议题，《庄子》文本的言意张力主要体现在语言的形式与内涵、心中意念与心外言语以及言语和达于天道这三个维度。第二，庄子并非完全反对一切言语，而是强调语言的界限，突出从“言不尽意”这一现实境遇出发，经由“得意忘言”的实现路径，最终朝向超越“言意”的达道境界这三重逻辑。这表面是不可知论，实则是可知论；第三，从言意观这一崭新视角切入“濠梁之辩”的文本，我们发现庄子对真意即“道”的把握并非是通过一般经验层面的感性、知性或理性认识方法的“以物观之”（惠子所做的），而是凭借非逻辑的直觉体悟和“天人合一”的超验体验即“以道观之”来实现的。介于文章篇幅所限，本文无法过多展开，但由此引出的以下三个问题值得进一步探讨：第一，从老子与庄子的角度；尽管二者在反对言说道立场上一致，但由于二者关切点的不同（宇宙论和人生哲学），对道家学派内关于体“道”方式，即老子“玄言”与庄子“三言”“非庄语”的细分差异值得进一步比较。第二，从儒家与庄子的角度；面对复杂的政治环境，孔子通过“微言大义”“春秋笔法”以一种“迂回”的方式来阐发圣人之意，而这与其所推崇的《周易》的“立象尽意”方法论关系密切。儒道两家似乎面对不可言都采取了相似而又不同的路径。同时，对庄子中的“意象”论也有待进一步分析。最后，从西方哲学与庄子的角度；无论是运用现代数理逻辑或者现代语言学“能指”和“所指”的区分来对庄子言意论证进行重构，还是探讨同为轴心时代的中国先秦语言与古希腊的“逻格斯”二者的分野，以及借鉴海德格尔与维特根斯坦对西方现代哲学语言的话语批判，并且分析由此带来的中西文化、思维方式差异的起源，都值得进一步思考。此外，正如张默生所说，“三言”是“庄子的钥匙” [23]。庄子的“三言”问题同样值得澄清。

参考文献

[1] 阮元校刻. 十三经注疏（清嘉庆刊本），周易正义 • 卷七 [M]，北京：中华书局，2009 年版，第 170-171 页。

[2] 邹晓丽. 基础汉字形义释源：《说文》部首今读本义 [M]. 中华书局, 1990 版，第 58 页。

[3] 此外，还有境界论上的“意境”，即人所处于的特定气质精神状态，庄子所谓体道或至一境界。详见陈波. 言意之辩: 诠释与评论 [J]. 江海学刊, 2005(3): 36-42.

[4] 王葆玹. 正始玄学 [M]. 齐鲁书社, 1987 年版，第 318 页。

[5] 陈鼓应. 老子今注今译 [M]. 北京：商务印书馆, 2006 年版，第 73、169 页。

[6] 陈鼓应. 庄子今注今译 [M]. 北京：商务印书馆, 2007 年版，第 668 页。以下引注简称《庄子》附页码。

[7]《庄子》，第 645 页。

[8]《庄子》，第 413 页。

[9]《庄子》，第 832-833 页。

[10] 郭庆藩. 庄子集释 [M]. 北京：中华书局，1961 年版，第 946 页。

[11]《庄子》，第 485 页。

[12]《庄子》，第 1016 页。

[13] 钟泰. 庄子发微 [M]. 上海：上海古籍出版社，2002 年版，第 791 页。

[14] 张学智. 先秦哲学中的名与意谓 [J]. 周易研究, 2021(06):83-90. “巵（卮）言”这一概念较为复杂，存在多种不同理解。“巵是酒器，郭注成疏由‘因物’‘从彼’立说，接近寓言，主要有两个涵义：一、巵对应于外的因物随变，二、巵所表示言说者的无主无执以至于无心。两义适所以相成。”参见徐圣心. 庄子“三言”的创用及其后设意义 [M]，台北花木兰文化出版社, 2009 年版，第 126 页。

[15] 王博. 庄子哲学 [M]. 北京：北京大学出版社，2004 年版，第 5 页。

[16] 冯友兰著. 三松堂全集第 6 卷 [M]. 郑州：河南人民出版社, 1989 年版，第 102 页。

[17]《庄子》，第 504 页。

[18]《庄子》，第 1019 页。

[19] 张文江.《庄子 • 天下篇》讲记 [J]. 上海文化, 2013(01):104-112.

[20] 郭庆藩. 庄子集释 [M]. 中华书局，1978，第 1100 页。

[21]《庄子》，第 265 页。

[22] 马丁 • 海德格尔著；孙周兴选编. 海德格尔选集 [M]. 上海：上海三联书店, 1996 年版，第 1055 页。

[23] 张默生:《庄子新释》，齐鲁书社，1993 年，第 12 页。

作者信息

焦成城　西安交通大学人文社会科学学院
西安 710049

珠峰学报 • 2024 春

钱院学辅公众号　qianyuanxuefu

钱院学辅信息站　https://qyxf.site

钱院学辅•书库　https://qyxf.site/latest

珠峰学报专栏　https://qyxf.site/zhufeng