同行专家业内评价意见书编号：20250854358

附件1

浙江工程师学院（浙江大学工程师学院）
同行专家业内评价意见书

姓名：谭浩

学号：22260090

申报工程师职称专业类别（领域）：电子信息

浙江工程师学院（浙江大学工程师学院）制

2025年03月13日

填表说明

一、本报告中相关的技术或数据如涉及知识产权保护、军工项目保密等内容，请作脱密处理。

二、请用宋体小四字号撰写本报告，可另行附页或增加页数，A4纸双面打印。

三、表中所涉及的签名都必须用蓝、黑色墨水笔，亲笔签名或签字章，不可以打印代替。

四、同行专家业内评价意见书编号由工程师学院填写，编号规则为：年份4位＋申报工程师职称专业类别(领域)4位+流水号3位，共11位。

一、个人申报

（一）基本情况【围绕《浙江工程师学院（浙江大学工程师学院）工程类专业学位研究生工程师职称评审参考指标》，结合该专业类别(领域)工程师职称评审相关标准，举例说明】

1.对本专业基础理论知识和专业技术知识掌握情况(不少于200字)

在智能科学与技术及计算机科学与技术领域展现了扎实的理论基础与专业技术能力。大学本科期间，其专业成绩位列前3/310，系统掌握了人工智能、机器学习、区块链等核心课程知识。硕士阶段进入浙江大学CAD&CG国家重点实验室，研究方向聚焦大语言模型（LLM）与智能体（Agent），深入研究了分布式系统、智能合约设计与合规性管理等前沿技术。

在专业技术知识方面，具备多领域交叉应用能力：
深度学习与语音处理：获得发明专利《基于定向降噪与干声提取技术的语音优化方法》，深入掌握语音信号建模、噪声场景分类、全卷积神经网络（FCNN）架构设计等技术。提出基于自监督学习的深度语音去噪框架，通过嵌入吸引子网络（DANet）实现高维时频信息分离，显著提升语音增强效果，并申请获得软著《声盾降噪助手》，体现对音频压缩、离散余弦变换（DCT）及人声分离算法的系统性掌握。

算法与模型优化：在大模型领域完成了Function Call模型优化，熟练应用数据增强、微调技术（如Lora微调等PEFT、全参微调），显著提升工具调用意图指标，此外一作发表相关论文：《Structured Knowledge Injection and Reasoning Prompting for Compliance On-chain Asset Analysis》展现了深度学习框架的实战能力。

区块链与智能合约：在“光伏资源通证化平台开发”项目中，主导链上身份管理系统设计，集成可验证凭证（VC）与合规引擎，攻克跨司法区合规规则动态适配难题，体现对分布式账本技术、零知识证明等理论的深入理解。并基于此以一作发表论文《FCToken: A Flexible Framework for Blockchain-Based Compliance Tokenization 》

多智能体系统（Multi-Agent）：在采购业务流程重构中，基于Multi-Agent框架将性能从0.78提升至0.93，并申请专利《一种基于大语言模型(LLM)和代理(Agent)技术的的需求导购助手》，展示了复杂系统设计与协同优化的专业知识。

此外获得多项国家级竞赛获奖（如中国大学生互联网+创新大赛金奖、挑战杯省级金奖等），进一步印证了在学术与工程领域的双重知识储备。

2.工程实践的经历(不少于200字)

在工程实践中积累了丰富的跨领域项目经验，涵盖算法优化、区块链开发与智能系统设计：
* AI模型优化与部署：在蚂蚁工作期间，构建首个动态/交互式Function Call数据集，开发数据构造框架，支持可控生成，并将模型性能提升至BFCL榜单10B以下模型第二（截止2024年8月），直接推动工具调用效率提升超30%。
*
区块链合规平台开发：在浙江舜宇智能光学技术有限公司专业实践期间，主导光伏资源通证化平台研发，设计链上身份管理系统与低代码智能合约生成工具，解决跨司法区合规难题，大大缩短合约开发周期，助力企业获得30万元项目经费支持。
*

智能合约与流程重构：在阿里部门工作期间，优化大规模特征数据处理流程，通过LoRA模型训练提升标注效率；在采购业务中基于Multi-
Agent框架重构流程，性能达标上线并申请专利，减少开发工程量40%。
其实践经历覆盖技术研发、系统设计、项目管理全流程，尤其在区块链与AI融合场景中展现了突出的工程落地能力。

3.在实际工作中综合运用所学知识解决复杂工程问题的案例（不少于1000字）

案例1:
项目背景：在浙江舜宇智能光学技术有限公司的“光伏资源通证化平台开发”项目中，需将光伏资源转化为链上合规数字资产，面临跨司法区合规管理、链上身份验证安全性、智能合约可扩展性三大技术难点。
问题复杂性分析：
1.
跨司法区合规规则动态适配：不同地区的法律差异导致传统合约难以灵活调整，需设计动态合规引擎。
2. 链上身份隐私与安全：用户身份需与区块链地址绑定，但需避免敏感信息泄露。
3.
低代码合约生成工具的可扩展性：现有工具难以支持复杂业务逻辑，影响平台响应速度。
解决方案与实施：
* 动态合规引擎设计：
    *
技术路线：基于智能合约开发可编程合规规则库，引入法律语义解析模块，将不同司法区条文转化为合约逻辑。
    *
实现方法：采用Solidity编写合规规则模板，支持实时更新；通过多签机制确保规则修改的合法性。
* 链上身份管理系统优化：
    *
技术路线：集成去中心化身份（DID）框架，采用可验证凭证（VC）与零知识证明（ZKP）。
    *
实现方法：用户通过权威机构完成KYC/AML认证，生成加密凭证并上链；交易时使用ZKP验证凭证有效性，避免暴露真实身份。
* 低代码智能合约生成工具开发：
    * 技术路线：构建可视化配置界面，基于领域特定语言（DSL）生成合约代码。
    *
实现方法：设计模块化合约组件库（如通证发行、交易清算），用户拖拽配置后自动生成并部署合约。
综合效益：
推动光伏资源数字化流转，助力碳中和目标；平台透明性提升投资者信心。
个人贡献：作为区块链开发负责人，谭浩主导了合规引擎与身份系统设计，提出并实施低代码工具开发方案，攻克技术难点3项，一作发表论文《FCToken: A Flexible Framework for Blockchain-Based Compliance Tokenization
》一篇，相关成果直接支撑其学位论文《基于可信能源数据的链上合规碳交易系统设计》。
[SEP]
案例2:

项目背景：在蚂蚁AI Native部门参与“Function Call模型优化与平台集成”项目，需解决工具调用意图识别率低、多任务并行处理能力不足问题，支撑芝士饼平台智能服务升级。

复杂性分析：

1. 动态工具调用场景适配：传统数据集缺乏交互式调用样本，模型难以处理动态参数传递与上下文依赖。
2. 多任务并行性能瓶颈：现有模型并行调用准确率仅68%，影响复杂任务执行效率。
3. 工程落地挑战：需将优化模型无缝集成至平台，同时确保推理延迟低于200ms。

解决方案与实施：

* 动态数据集构建：
    * 技术路线：设计Multi-Agent仿真框架，模拟用户与工具交互场景，生成包含动态参数、嵌套调用的多样化数据。
    * 实现方法：基于Python构建自动化数据生成管道，引入规则引擎控制Agent行为逻辑，生成10万+高质量样本，覆盖80%常见工具调用场景。
* 模型微调与并行优化：
    * 技术路线：采用Lora微调方案，优化qwen2-7b与72b模型，提升其工具调用能力。
    * 实现方法：引入分层注意力头，分离工具选择与参数生成任务；通过课程学习策略逐步增加并行任务复杂度。

个人贡献：作为项目负责人，主导技术方案设计并推动落地，一作发表论文一篇：《Structured Knowledge Injection and Reasoning Prompting for Compliance On-chain Asset Analysis》

（二）取得的业绩（代表作）【限填3项，须提交证明原件（包括发表的论文、出版的著作、专利证书、获奖证书、科技项目立项文件或合同、企业证明等）供核实，并提供复印件一份】

1.
公开成果代表作【论文发表、专利成果、软件著作权、标准规范与行业工法制定、著作编写、科技成果获奖、学位论文等】

成果名称	成果类别 [含论文、授权专利（含发明专利申请）、软件著作权、标准、工法、著作、获奖、学位论文等]	发表时间/授权或申请时间等	刊物名称/专利授权或申请号等	本人排名/总人数	备注
FCToken: A Flexible Framework for Blockchain-Based Compliance Tokenization	会议论文	2024年02月06日	DOI: 10.1109/ICDMW60847.2023.00093	1/6	
基于定向降噪与干声提取	发明专利申请	2021年05月27日	申请号：202110587258.6	1/5	

2.其他代表作【主持或参与的课题研究项目、科技成果应用转化推广、企业技术难题解决方案、自主研发设计的产品或样机、技术报告、设计图纸、软课题研究报告、可行性研究报告、规划设计方案、施工或调试报告、工程实验、技术培训教材、推动行业发展中发挥的作用及取得的经济社会效益等】

一作录用论文：《Structured Knowledge Injection and Reasoning Prompting for Compliance On-chain Asset Analysis》。收录于会议：IJCAI-OpenKG。

（三）在校期间课程、专业实践训练及学位论文相关情况	
课程成绩情况	按课程学分核算的平均成绩： 82 分
专业实践训练时间及考核情况(具有三年及以上工作经历的不作要求)	累计时间： 1.2 年（要求1年及以上） 考核成绩： 84 分
本人承诺	
个人声明：本人上述所填资料均为真实有效，如有虚假，愿承担一切责任，特此声明！ **申报人签名：**	

2226090

二、日常表现考核评价及申报材料审核公示结果

日常表现考核评价	非定向生由德育导师考核评价、定向生由所在工作单位考核评价： ☐优秀 ☐良好 ☐合格 ☐不合格 德育导师/定向生所在工作单位分管领导签字（公章）：　　年　月　日
申报材料审核公示	根据评审条件，工程师学院已对申报人员进行材料审核（学位课程成绩、专业实践训练时间及考核、学位论文、代表作等情况），并将符合要求的申报材料在学院网站公示不少于5个工作日，具体公示结果如下： ☐通过 ☐不通过（具体原因：　　） 工程师学院教学管理办公室审核签字（公章）：　　年　月　日

浙 江 大 学 研 究 生 院

攻读硕士学位研究生成绩表

学号：22260090	姓名：谭浩	性别：男	学院：工程师学院	专业：计算机技术	学制：2.5年
毕业时最低应获：26.0学分	已获得：28.0学分			入学年月：2022-09	毕业年月：
学位证书号：		毕业证书号：			授予学位：

学习时间	课程名称	备注	学分	成绩	课程性质	学习时间	课程名称	备注	学分	成绩	课程性质
2022-2023学年秋季学期	新时代中国特色社会主义理论与实践		2.0	87	专业学位课	2022-2023学年春季学期	自然辩证法概论		1.0	74	专业学位课
2022-2023学年秋季学期	工程技术创新前沿		1.5	87	专业学位课	2022-2023学年夏季学期	研究生英语基础技能		1.0	免修	公共学位课
2022-2023学年秋季学期	数值计算方法		2.0	88	专业选修课	2022-2023学年夏季学期	物联网信息安全技术与应用基础		2.0	83	专业选修课
2022-2023学年秋冬学期	高阶工程认知实践		3.0	76	专业学位课	2022-2023学年夏季学期	工程师创新创业思维		2.0	96	专业选修课
2022-2023学年秋冬学期	研究生论文写作指导		1.0	71	专业选修课	2022-2023学年夏季学期	研究生英语		2.0	免修	专业学位课
2022-2023学年冬季学期	产业技术发展前沿		1.5	86	专业学位课	2022-2023学年春夏学期	移动互联网智能设备应用设计与实践		3.0	85	专业学位课
2022-2023学年冬季学期	物联网操作系统与边缘计算		2.0	86	专业选修课		硕士生读书报告		2.0	通过	
2022-2023学年春季学期	工程伦理		2.0	64	专业学位课						

说明：1. 研究生课程按三种方法计分：百分制，两级制（通过、不通过），五级制（优、良、中、及格、不及格）。

2. 备注中“*”表示重修课程。

学院成绩校核章：

成绩校核人：张梦依

打印日期：2025-03-20

经检索“Engineering Village”，下述论文被《Ei Compendex》收录。（检索时间：2024 年 12 月 23 日）。

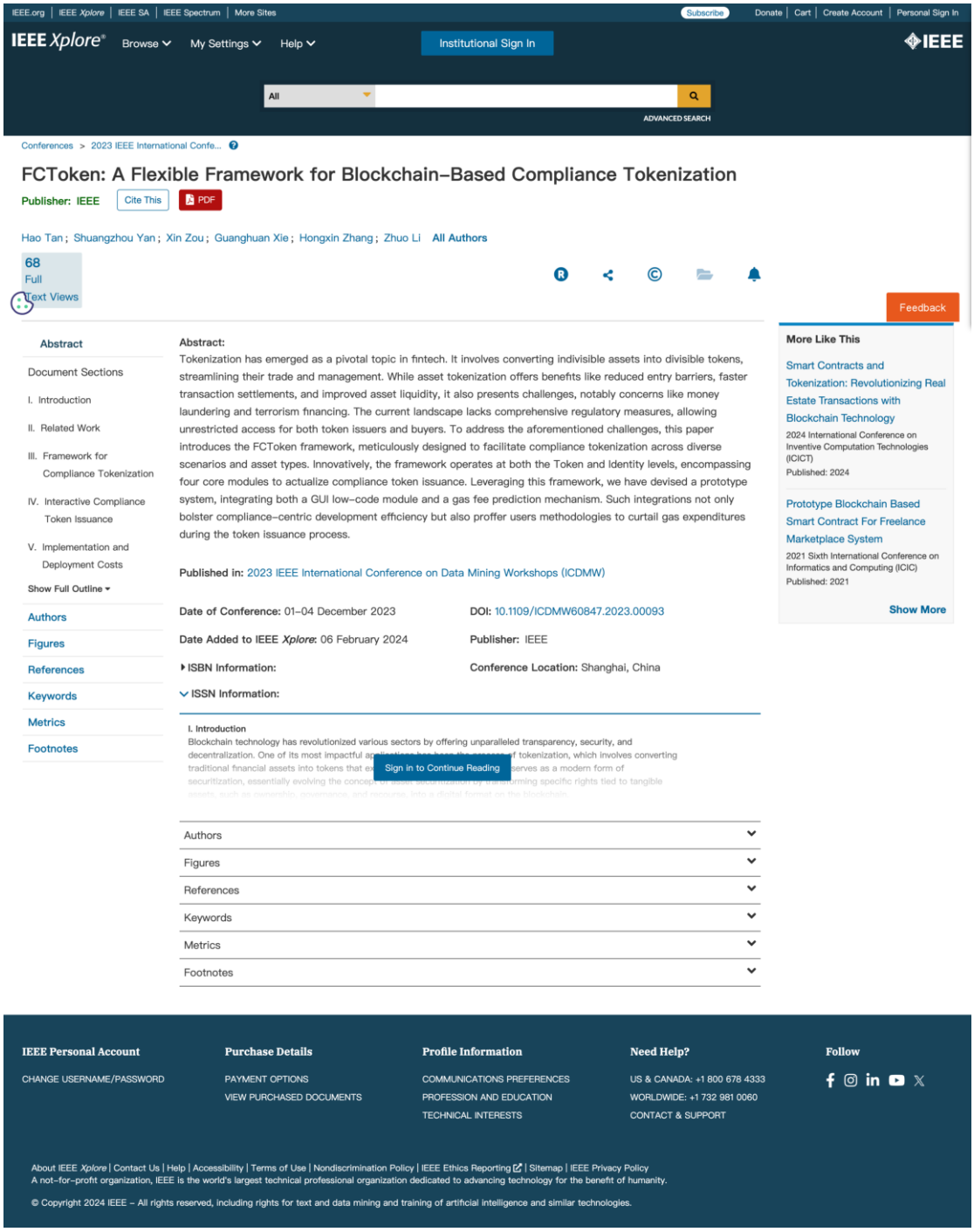
<RECORD 1>
Accession number:20240915663273
Title:FCToken: A Flexible Framework for Blockchain-Based Compliance Tokenization
Authors:Tan, Hao (1); Yan, Shuangzhou (1); Zou, Xin (1); Xie, Guanghuan (1); Zhang, Hongxin (1); Li, Zhuo (2)
Author affiliation:(1) Zhejiang University, State Key Lab of Cad&cg, Hangzhou, China; (2) State Street Technology Ltd., Global Technology Service Department, Hangzhou, China
Corresponding authors:Zhang, Hongxin(zhx@zju.edu.cn); Li, Zhuo(lizhuo@zju.edu.cn)
Source title:IEEE International Conference on Data Mining Workshops, ICDMW
Abbreviated source title:IEEE Int. Conf. Data Mining Workshops, ICDMW
Part number:1 of 1
Issue title:Proceedings - 23rd IEEE International Conference on Data Mining Workshops, ICDMW 2023
Issue date:2023
Publication year:2023
Pages:671-681
Language:English
ISSN:23759232
E-ISSN:23759259
ISBN-13:9798350381641
Document type:Conference article (CA)
Conference name:23rd IEEE International Conference on Data Mining Workshops, ICDMW 2023
Conference date:December 1, 2023 - December 4, 2023
Conference location:Shanghai, China
Conference code:197121
Sponsor:IEEE Computer Society; Technology Innovation Institute; TWO SIGMA; US National Science Foundation (NSF)
Publisher:IEEE Computer Society
Number of references:26
Main heading:Blockchain
Controlled terms:Codes (symbols)
Uncontrolled terms:'current - Block-chain - Compliance - Entry barriers - Flexible framework - Four-core - Gas fee - Low-code - Prototype system - Tokenization
Classification code:723.2 Data Processing and Image Processing - 723.3 Database Systems
DOI:10.1109/ICDMW60847.2023.00093
Database:Compendex

注：
1. 以上检索结果来自 CALIS 查收查引系统。
2. 以上检索结果均得到委托人及被检索作者的确认。

教育部科技查新工作站（Z09）
检索人（签章）：刘琪琪
2024 年 12 月 31 日

原文链接：https://ieeexplore.ieee.org/document/10411565

DOI：10.1109/ICDMW60847.2023.00093



(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 113314136 A
(43)申请公布日 2021.08.27

(21)申请号 202110587258.6

(22)申请日 2021.05.27

(71)申请人 西安电子科技大学
地址 710126 陕西省西安市西安电子科技大学长安校区

(72)发明人 谭浩 刘天翼 郭哲宇 郝佳晨 樊书宏

(74)专利代理机构 北京神州信德知识产权代理事务所(普通合伙) 11814
代理人 刘真

(51)Int.Cl.
G10L 21/0208(2013.01)
G10L 25/30(2013.01)
G10L 25/60(2013.01)

权利要求书2页 说明书6页 附图2页

(54)发明名称

基于定向降噪与干声提取技术的语音优化方法

(57)摘要

本发明是一种基于定向降噪与干声提取技术的语音优化方法，该方法包括以下步骤：S1、建立声音库；S2、定义使用的环境；S3、构建深度语音去噪自监督语音增强全卷积神经网络；S4、进行声音增强。本发明能够针对餐厅、室外、马路等不同的场景进行降噪，大大提升了降噪的效果。

CN 113314136 A

Fwd: [LKM@IJCAI2024] Notification for Paper Proceedings & Presentation

tttat于7月2日 12:44发给ysz22。

完整信息

发件人: Microsoft CMT <email@msr-cmt.org>
日期: 2024年6月29日 GMT+8 17:26:59
收件人: hao tan <tttat@zju.edu.cn>
主题: **[LKM@IJCAI2024] Notification for Paper Proceedings & Presentation**

Dear hao tan,

Congratulations again for the acceptance of your paper:

16 Structured knowledge injection and reasoning prompting for compliance on-chain asset analysis

at LKM2024: The First International OpenKG Workshop Large Knowledge-Enhanced Models @IJCAI 2024.

For archival paper proceedings:

Apologies for the delay in coordinating with the journal. We will soon open a submission channel for the recommended journal (Data Intelligence). The papers will undergo rapid peer review and be published in the proceedings (EI index).

For non-archival paper proceedings:

Please send us an online link (e.g., ArXiv) of your paper, and we will put the link on the website.

For presentation:

1.Registration can be workshop only.

2.Papers can be presented in two formats: poster or oral presentation. We will send the notification of poster and oral selection in the next week. We recommend that papers selected for oral presentation also prepare a poster to facilitate discussion.

3.Oral presentation is 10min for each paper, we recommend to present onsite, detailed schedule will be online in the next week at https://lkm2024.openkg.org/.

4. The poster can be any format, we recommend to use the size "24 inches wide x 36 inches high" in portrait orientation. Posters can be picked up and set up at 10:30-11:00 & 11:30-12:00 & 15:30-16:00 in the workshop rooms.

In LKM@IJCAI2024, we will have keynotes & invited talks from the University of Texas at Austin, Tsinghua University, Southeast University, Beijing University of Posts and Telecommunications and Ant Group.

Looking forward to meeting you in Jeju, South Korea, August 3!

Any questions, contact: zhangningyu@zju.edu.cn

Sincerely,

LKM 2024 Program Chairs

Structured Knowledge Injection and Reasoning Prompting for Compliance On-chain Asset Analysis

Hao Tan[1*], **Shuangzhou Yan**[1*], **Hongxin Zhang**[1†], **Zhuo Li**[2†]
[1]Zhejiang University
[2]State Street Technology (Zhejiang) Ltd.
{tttat, ysz22, zhx, lizhuo}@zju.edu.cn,

Abstract

The integration of Domain-Specific Languages (DSL) with Large Language Models (LLM) offers an innovative approach to addressing complex problems within specific domains. This paper proposes the method of structured knowledge injection and reasoning prompting, and conducts experimental analysis on the task of compliance on-chain asset analysis. DSL, as a specialized form of knowledge representation, is tightly coupled with domain-specific terminology, allowing expert knowledge to be presented in a structured form. Leveraging the advantages of DSL in knowledge representation, we propose its combination with LLMs to enhance the model's capability to handle complex tasks within specific domains. In this paper, we provide a detailed explanation of our method's application to solving the complex task of on-chain compliance asset analysis. Compared to traditional LLM prompting methods, our approach shows significant improvements in key metrics.

1 Introduction

Recently, there has been notable advancement in the field of natural language processing thanks to the development of large language models (LLMs). LLMs are commonly trained on large volumes of internet text data, allowing them can easily adapt to diverse tasks in various domains without needing specialized data for each task.

The design of prompts is essential in the process of adapting pre-trained LLMs to new tasks with limited or no task-specific training data. By employing prompt engineering techniques, these models are able to outperform in a wide range of jobs and fields. In contrast to traditional paradigms, which usually need model retraining or substantial fine-tuning to obtain performance on specific tasks, this adaptability stands out

Analysis of smart contracts for on-chain compliance tokens is a real problem in the production process. On-chain financial assets typically possess hundreds of financial attributes and various compliance rule clauses. Unlike traditional financial products, financial domain experts analyzing on-chain compliance assets need to read and understand Solidity code and combine it with financial domain knowledge to analyze on-chain financial products. This task comprises multiple sub-tasks such as understanding Solidity code, extracting financial attributes, and reasoning about different compliance clauses. Indeed, in real-life situations, there are numerous domain-specific tasks that have similarities and are characterized by complexity and strict limitations. In order to tackle these issues, we suggest implementing a method called structured knowledge injection and reasoning prompt for LLMs. This approach entails the injection of DSL syntax into LLMs and the utilization of specialized syntax and DSL to facilitate the handling of intricate restrictions and task solutions by LLMs.

The justification for utilizing DSL is grounded in the research conducted by [Decker, 1998] and [Seipel *et al.*, 2018]. These scholars propose that DSL, with their syntax closely aligned with domain-

[*]Both authors contributed equally to this research.
[†]Corresponding authors.

specific terminology, offer a more adaptable means of representing expert knowledge. DSL function as a means of KR (knowledge representing) and also improve the ability to execute knowledge, enabling the organization and formalization of unstructured knowledge. Given the benefits of DSL in organizing knowledge and handling intricate restrictions, we suggest integrating DSL with LLMs.

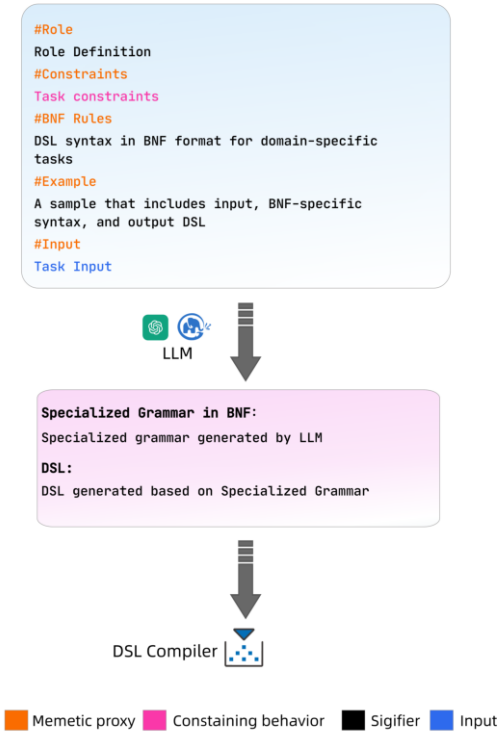


Figure 1: Architecture of "structured knowledge injection and reasoning prompting"

Our approach involves encoding domain knowledge as a DSL and designing its syntax using the Backus-Naur Form (BNF)[McCracken and Reilly, 2003] representation. LLM incorporates structured knowledge, specifically the BNF syntax paradigm, throughout task execution. A specialized subset of BNF syntax is utilized as intermediate reasoning steps and restrictions. Ultimately, we generate the outcome in DSL format, which is then handed over to an external DSL compiler for the purpose of parsing and resolving certain issues. The structure of our strategy is depicted in Figure 1.

In order to evaluate the enhancement in performance of our method in problem-solving, we generated a dataset specifically designed for analyzing on-chain compliance token smart contracts. The dataset comprises 200 compliance token smart contracts, each representing a different type. Each contract corresponds to an on-chain asset that possesses a minimum of 30 distinct financial qualities. Additionally, there are 20 unique compliance rules and at least 8 user admission conditions within each contract. Furthermore, each contract is composed of at least four sub-contracts. We have developed five distinct measures to evaluate the quality of the outcomes produced by LLM, including both broad and detailed viewpoints.

In particular, we investigated the utilization of GPT models gpt-3.5-turbo, gpt-4 and ChatCLM model chatglm3-turbo for analyzing on-chain assets, and we compared their performance against different reference points. Empirical findings demonstrate that our approach substantially improves the capacity of all algorithms to assess on-chain compliance token smart contracts. The effectiveness of our approach showcases the immense capability of utilizing the inherent information within LLMs for intricate task reasoning, achieved through a meticulously crafted fundamental structure and prompts specifically tailored for LLMs.

This achievement demonstrates that the application of structured knowledge injection and reasoning prompt approach for LLMs is highly effective in resolving intricate difficulties within specified domains.

2 Blockchain, Solidity and Compliance Token

Blockchain is a distributed ledger technology that connects data through blocks, creating an immutable, transparent, and decentralized database [Bhutta *et al.*, 2021]. Blockchain technology ensures data consistency and trustworthiness through a consensus mechanism.

Smart contracts represent automated contracts executed on the blockchain. They facilitate trustworthy, secure, and transparent transactions and contract executions. Smart contracts are written using Solidity, a high-level programming language that is specifically designed for creating smart contracts and is widely used on Ethereum. Solidity can define the state and behavior of contracts and interact with other contracts. It is the most commonly used language for writing smart contracts at present.

Compliance tokens are a form of regulated securities issued using blockchain technology. These tokens can represent shares, bonds, real estate, and other real assets, traded and held via the blockchain. Examining compliance tokens and their corresponding smart contracts is essential in contemporary financial and technical settings.Compliance analysis assists firms in discovering and capitalising on emerging blockchain and smart contract prospects while complying with rules, thereby facilitating worldwide expansion and fostering innovation.

3 Related Work

GPT-3[Brown *et al.*, 2020] demonstrated a strong capability to perform few-shot predictions, where the model is given a description of the task in natural language with few examples. Scaling model size, data, and computing are crucial to enable this learning ability.[Rae *et al.*, 2021; Smith *et al.*, 2022; Chowdhery *et al.*, 2023; Dua *et al.*, 2019] have recently suggested training diverse types of LLMs using distinct training techniques. Smaller language models lack the capability to tackle new tasks by learning from a small number of examples. This ability only becomes apparent when the size of the model is raised, as stated by [Kaplan *et al.*, 2020]. In recent times, a number of studies [Xie *et al.*, 2023; Work,]) have focused on comprehending the mechanisms and rationales behind in-context learning. Another study is BINDER [Cheng *et al.*, 2022], which employs Codex to generate ”soft” SQL queries for responding to inquiries based on tables.

Chain-of-thought prompting: Although LLMs have shown significant success in various natural language processing tasks, their reasoning capabilities are often seen as a limitation. Recently, Chain-of-Thought (CoT) reasoning[Wei *et al.*, 2022; Kojima *et al.*, 2022; Wang *et al.*, 2022] has been proposed to enhance LLMs' ability to perform reasoning tasks by demonstrating the ”reasoning process in natural language.” [Suzgun *et al.*, 2022]have shown that CoT can surpass human performance on challenging BIG-Bench tasks. Subsequently, several other studies [Drozdov *et al.*, 2022; Nye *et al.*, 2021] have proposed different methods to leverage LLMs for reasoning tasks by allowing intermediate steps. ReAct[Yao *et al.*, 2022] suggests enhancing LLMs' reasoning capabilities by utilizing external tools such as search engines. [Chen *et al.*, 2022] have proposed ”Program of Thoughts” (PoT), a novel prompting method that generates program

Figure 2: Syntactic Definition of our DSL

code to express reasoning steps and delegates computational tasks to a program interpreter, effectively separating the reasoning and computation processes.

LLMs for program generation and semantic parsing: Generating programs from natural language specifications, often referred to as semantic parsing, is a sub-problem of program synthesis. [Austin *et al.*, 2021; Xu *et al.*, 2022] have explored methods for using LLMs to generate code in general programming languages (e.g., Python). There is also work on using LLMs for tool usage through further training[Schick *et al.*, 2024] or prompting[Qin *et al.*, 2023; Wang *et al.*, 2023] , studying how model size[Qiu *et al.*, 2022] and retrievers[Ye *et al.*, 2020; Levy *et al.*, 2022] affect in-context learning, semantic parsing, and constrained decoding in program generation[Scholak *et al.*, 2021; Poesia *et al.*, 2022].

4 Methodology

4.1 Design Principles and Syntax of DSL

Our approach uses LLM to collect and analyze on-chain token issuance data, identifying key frameworks and trends without domain experts. LLM's language and pattern recognition capabilities enable efficient analysis of complex on-chain data.

For data labeling, we use top market cap tokens and NFTs from Etherscan and Cryptoslam, respectively, resulting in 975 smart contracts and 14,200 functions after data cleansing. We analyze these contracts to identify common features and components for DSL design.

We process the source code using GPT-4[Achiam *et al.*, 2023] and standardize outputs through prompt

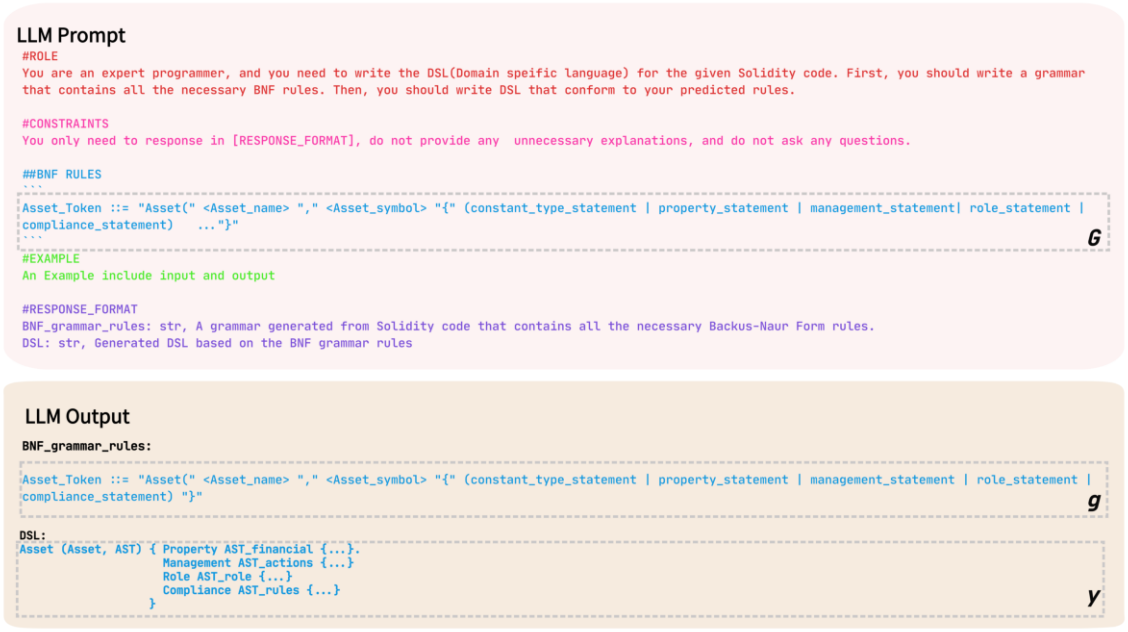


Figure 3: Examples of our prompting method. G represents for the entire DSL grammar, g represents for the specific subset of grammar and y represents for the matching specialized DSL

engineering. A JSON label structure captures code structure, roles, functions, and compilation versions. Analyzing 975 smart contracts, we identify 2,447 unique functions after removing duplicates. Using vectorization and DBSCAN clustering, we categorize the top 20 frequent functions into five categories: Token Management, Permission Control, Contract Upgrading, Contract Framework Construction, and Mathematical Calculation. Essential functions include transfer, transferFrom, approve, balanceOf, allowance, totalSupply, burn, mint, transferownership, and constructor.

We specify the grammar in our DSL using the BNF. Figure 2 illustrates the syntactic definition of our DSL. Compliance assets, designated as **Asset**, have a structure similar to a class in object-oriented programming. When designing an Asset, the name and symbol must be specified. Each Asset includes the setup of static variables and the execution of several modules, such as:

- **Type System**: Includes Claim and Token. Claim represents qualifications like compliance standards. We manage a database for setting and assigning Claim details in DSL. Token is similar to the address type in Solidity and is used for ERC20 asset configuration.
- **Property**: Includes financial asset details such as admittance fee or notional price. Time-related attributes are represented by the *date* variable.
- **Management**: Manages on-chain asset behavior, defining actions like asset purchase and margin calls. Key actions include *Trans*, *TransF*, *Mintable*, and *Burnable*.
- **Role**: Represents participants in the compliance token market, setting regulatory limitations and specifying eligible groups.
- **Compliance**: Involves adding compliance clauses to the asset. A compliance clause includes a *role condition, asset condition, time condition*, and *action*, defining when specific actions can occur.

4.2 Thought with Specific Knowledge Representation

To address intricate challenges, we propose a novel approach called Prompt Engineering based on Structural Knowledge Injection and Reasoning, which

enhances the learning process in the LLM environment. In order to improve LLM's proficiency in numerical reasoning, PoT transforms the logical processes of reasoning into Python code that can be executed, effectively separating the process of reasoning from the computational aspect of solving mathematical issues. SCoT aims to transform the intermediate reasoning processes of CoT into structured source code, which allows LLM to produce code with greater precision. The primary aim of the study is to enhance LLM's capabilities in numerical computation and code generation using the GPL(general-purpose programming language), specifically the Python scripting language. Nevertheless, when confronted with intricate practical issues, these solutions are not suitable.

However, drawing inspiration from these approaches, given that the introduction of General Purpose Languages (GPL) is insufficient for solving the majority of jobs, it would be advantageous to consider implementing DSL tailored to specific tasks in order to efficiently address intricate difficulties. According to Wang et al.'s findings in Grammar Prompt, LLM demonstrates exceptional proficiency in both utilizing and creating DSL. It can showcase strong capabilities for many DSL.

Therefore, we propose a Prompt method: structured knowledge injection and reasoning prompting.Instead of using PoT/SCoT, our approach involves creating customized DSL and their related DSL grammars tailored to specific activities. During the process of reasoning in LLM, we include the entire DSL grammar G into the context of LLM. Meanwhile, during the output phase of the LLM, we allow LLM to make predictions for a specific subset of grammar, called g, based on the user's request. This subset is a part of the set G. Subsequently, with the anticipated subset g, LLM generates the matching specialized DSL y. Ultimately, we employ a DSL encoder S to analyze y and acquire the ultimate outcome of the task. The method's structure is depicted in Figure 1. Our method primarily consists of four components: DSL Grammar Injection under BNF Paradigm, Specialized Grammar Generation, DSL Generation, and Examples. An example of a Prompt is illustrated in Figure 3.

Our approach strictly follows the prompt principles outlined by Reynolds and McDonell (2021), which encompass the following: Signifier: A signifier is a pattern which keys the intended behavior. Memetic proxy: A memetic proxy is a concept in which a character or characteristic situation is used as a proxy for an intention. Constraining behavior: In addition to directing the LLM on the desirable response. Meta prompting: A meta prompt is a short phrase or a fill-in-the-blank template encapsulat-ing a more general intention that will unfold into a more specific prompt when combined with ad-ditional information. First, we require the LLM to assume the role of a programming expert and define its task. Subsequently, we impose limitations on the LLM's conduct within the given Prompt. These constraints are unrelated to the DSL Grammar and are typically designed according to the specific task context. Next, we incorporate the BNF paradigm DSL grammar, typically opting for the entire DSL grammar to empower the LLM in producing a more precise specialized grammar subset. To further improve the LLM's compliance with instructions, we incorporate illustrations.

Table 1: Contract Categorization Rules

Level	Number of Financial Attributes
Simple	30-70
Medium	71-110
Difficult	111-150

5 Experiments

5.1 Experiment set up

Dataset: In our analysis of compliance token smart contracts, we have compiled a corresponding dataset. This dataset consists of 200 on-chain compliance token smart contracts with different levels of complexity. It also includes their corresponding DSL code. The purpose of this dataset is to evaluate the LLM's ability to analyze on-chain compliance tokens. The complexity of the smart contracts in the collection is determined by the quantity of financial attributes. The analysis difficulty of these compliance contracts has been classified into three levels: simple, medium, and difficult. Table 1 outlines the criteria used to classify the contracts into different levels.

We conducted an experiment to evaluate the effectiveness of several methods in completing tasks of different levels of complexity.

Baseline: Our findings comprise three different models, including gpt-3.5-turbo, gpt-4, and chatglm3. We examine two prediction strategies: the normal prompt technique, which produces results by specifying the output format, called STD in short and the chain of thought prompt approach,

which generates results through sequential reasoning, called COT in short.

Implementation Details: For our experiment, we predominantly utilized the OpenAI API and the ChatGLM API. Our approach involved utilizing the LLM to generate DSL code, which was subsequently processed by an external DSL parsing tool to generate the ultimate compliance asset analysis report. The LLM output for the STD and COT techniques was provided in a predefined JSON format.

In order to accommodate few-shot scenarios, we have incorporated specific example types for various procedures. Our method involves utilizing input Solidity code, specialized DSL Grammar for the input code, and generating DSL code based on the DSL Grammar. The STD and COT methods supplied examples that consisted of Solidity code as input and the matching JSON format as output. Employing enhanced output instructions aids the model in comprehending the designated output format more effectively. Thus, in order to strengthen the output requirements of the LLM, we included a #RESPONSE FORMAT prompt for each method to provide a consistent format for the LLM's output.

Metric Design:

For the task of analyzing compliance token smart contracts, we have designed detailed metrics to evaluate performance from both macro and micro dimensions. The definitions and calculation formulas for each measure are as follows:

- **FAA**: Financial Attribute Accuracy (Macro)
- **KCAR**: KYC Clause Accuracy Rate (Macro)
- **UACIA**: User Admission Clause Identification Accuracy (Micro)
- **ARCIA**: Asset Requirement Clause Identification Accuracy (Micro)
- **TRICA**: Time-Related Clause Identification Accuracy (Micro)

5.2 Main Result

Main Results Table 2 presents our results on the complete dataset, while Tables 2-4 showcase comparisons of various methods across different models on datasets of varying difficulty levels. In terms of the KCAR metric, for the gpt-3.5-turbo model, our method increased the number of correctly identified instances from nearly **300** to over **2100**; for the gpt-4 model and the chatgLM-3-turbo model, performance improved by **15.0%** and **48.5%**, respectively. The significant enhancements on GPT-3.5 and ChatGLM stem from the nature of the KCAR metric, which requires LLMs to extract user qualification numbers from Solidity smart contract code and cross-reference them with a provided qualification list. Our method effectively determines qualification numbers based on DSL syntax constraints and accurately corresponds qualification clauses by leveraging a DSL compilation tool, resulting in a marked performance improvement. For the UACA metric, our method compared to the other two methods achieved improvements of over **87.3%**/ranking first/**58.5%**. Regarding the ARICA task, our method demonstrated significant performance gains. The baseline methods failed to accurately identify any compliance clauses related to assets in gpt-3.5-turbo and gpt-4 models, whereas our method successfully identified **28.7%** and **55.7%** of the data, respectively. Due to the performance limitations of chatglm3-turbo, our method's improvement was modest, with an accuracy of 0.4% compared to CoT's 0.3%. This discrepancy arises because the ARICA task involves extracting compliance clauses related to user-held assets, including the recognition of fund amounts and currency types. Many on-chain asset currencies may not be present in the LLM's pre-training data, such as newly introduced tokens like MNT, FET, and PEPE. As a result, LLMs may fail to recognize them. However, our method can accurately identify asset quantities and currency types through DSL syntax constraints.

Omission in responses For the FAA and TRCIA metrics, our method did not achieve notable improvements. Consequently, we conducted a more detailed analysis of the experimental data.

As introduced in Section 5.1, we categorized the data in our dataset into three levels of complexity: simple, medium, and difficult. We utilized the gpt-3.5-turbo model to analyze the performance across these three dataset levels, with results shown in Table 3.

It is evident that our method achieved the best results across all metrics in the simple-level data, with significant performance improvements. However, for medium and difficult levels, our method underperformed compared to the other two methods in the FAA metric (financial attribute identification) and the TRCIA metric (time compliance clause identification). Analysis of the LLM outputs revealed the issue: when handling large data volumes, the models exhibited "lazy" behavior, using "..." to replace complete answers, leading to incomplete outputs. Approximately 20% of the responses showed such

Table 2: Experiment result for different method and different model

Config	FAA	KCAR	UACA	ARICA	TRICA
G3.5-STD	**0.795**	0.121	0.324	0.006	0.186
G3.5-COT	0.790	0.114	0.293	0.006	0.212
G3.5-Ours	0.784	**0.686**	**0.607**	**0.288**	0.186
G4-STD	0.746	0.505	0.448	0.000	0.469
G4-COT	**0.767**	0.468	0.638	0.001	**0.735**
G4-Ours	0.571	**0.580**	**0.638**	**0.557**	0.548
C3-STD	**0.792**	0.235	0.224	0.000	**0.257**
C3-COT	0.787	0.230	0.115	0.003	0.086
C3-Ours	0.366	**0.349**	**0.355**	**0.004**	0.207

Note: G3.5 denotes GPT-3.5, G4 denotes GPT-4, and C3 denotes ChatGLM-3.

omissions, with only 3% occurring in simple data and over 30 instances in medium and difficult data. This is due to the complexity of generating BNF DSL syntax outputs for our method; complex data outputs led LLMs to perceive data types as repetitive, resulting in self-omissions.

Zero-shot Results Lastly, we evaluated the zero-shot performance of our method, comparing it with one-shot in Table 4. It is clear that our method heavily relies on sample setups; the absence of samples significantly degrades performance. Despite providing DSL syntax in the outputs, the relationship between DSL syntax and its conversion remains complex, necessitating designed examples to prompt conversion rules effectively.

Table 3: Performance Comparison of GPT-3.5-turbo Across Different Dataset Levels

Method	FAA	KCAR	UACA	ARICA	TRICA
Simple					
STD	0.797	0.102	0.325	0.047	0.196
COT	0.807	0.128	0.268	0.006	0.193
Ours	**0.942**	**0.710**	**0.720**	**0.349**	**0.212**
Medium					
STD	**0.811**	0.128	0.346	0.011	**0.214**
COT	0.797	0.116	0.282	0.005	0.183
Ours	0.751	**0.696**	**0.636**	**0.280**	0.194
Difficult					
STD	0.784	0.140	0.304	0.001	0.154
COT	**0.784**	0.096	0.330	0.008	**0.256**
Ours	0.734	**0.648**	**0.454**	**0.225**	0.150

Table 4: Experiment Results Comparing Zero-shot and One-shot Performance of Our Method Using GPT-3.5

Config	FAA	KCAR	UACA	ARICA	TRICA
Zero-shot	0.715	0.000	0.507	0.275	**0.330**
One-shot	**0.784**	**0.686**	**0.607**	**0.288**	0.186

Table 5: Results for molecule generation

Method	V	D	R	M
LLM with DSL	98	0.74	91.0	93.3
Standard Prompting	87.7	0.73	80.0	76.7

Note: The metrics are validity (V), diversity (D), retrosynthesis score (R) and membership (M). Higher is better for all metrics.

5.3 More Than Solidity Code Analysis

Ultimately, our objective is to prove, using appropriate experimental data, that our method can be used not only for code analysis jobs but also for a wide range of other activities with favorable outcomes. As an illustration, we chose the problem of producing distinct kinds of molecules. The empirical data is derived from Grammar Prompting[Wang *et al.*, 2024]. The result on molecule class belongs to Acrylates was shown in Table 5

6 Conclusion and Future Work

This work explores the utilization of DSL and DSL syntax as structural knowledge to augment the problem-solving capabilities of LLMs for complicated tasks. In the task of compliance token analysis, our method demonstrates a leading advantage across multiple metrics.DSL,and its syntax, which is described in Backus-Naur Form, are utilized as specific techniques for representing knowledge. They not only act as tools for knowledge representation but also enhance the executability of knowledge, allowing unstructured knowledge to be structured and formalized. This enhances the adaptable depiction of knowledge appropriate to a certain domain. [Austin *et al.*, 2021] introduced a framework for DSL that is based on ontology. This framework enables the DSL to undergo dynamic modifications, thus accommodating advancements. In the future, we will investigate the incorporation of dynamic DSL with LLMs to consistently enhance the problem-solving capacities of LLMs.

References

[Achiam *et al.*, 2023] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[Austin *et al.*, 2021] Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

[Bhutta *et al.*, 2021] Muhammad Nasir Mumtaz Bhutta, Amir A. Khwaja, Adnan Nadeem, Hafiz Farooq Ahmad, Muhammad Khurram Khan, Moataz A. Hanif, Houbing Song, Majed Alshamari, and Yue Cao. A survey on blockchain technology: Evolution, architecture and security. *IEEE Access*, 9:61048–61073, 2021.

[Brown *et al.*, 2020] TB Brown, B Mann, N Ryder, M Subbiah, JD Kaplan, P Dhariwal, A Neelakantan, P Shyam, G Sastry, A Askell, et al. Language models are few-shot learners advances in neural information processing systems 33. 2020.

[Chen *et al.*, 2022] Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W Cohen. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *arXiv preprint arXiv:2211.12588*, 2022.

[Cheng *et al.*, 2022] Zhoujun Cheng, Tianbao Xie, Peng Shi, Chengzu Li, Rahul Nadkarni, Yushi Hu, Caiming Xiong, Dragomir Radev, Mari Ostendorf, Luke Zettlemoyer, et al. Binding language models in symbolic languages. *arXiv preprint arXiv:2210.02875*, 2022.

[Chowdhery *et al.*, 2023] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113, 2023.

[Decker, 1998] Stefan Decker. On domain-specific declarative knowledge representation and database languages. In *KRDB-98—Proceedings of the 5th Workshop Knowledge Representation meets DataBases, Seattle, WA*, 1998.

[Drozdov *et al.*, 2022] Andrew Drozdov, Nathanael Schärli, Ekin Akyürek, Nathan Scales, Xinying Song, Xinyun Chen, Olivier Bousquet, and Denny Zhou. Compositional semantic parsing with large language models. In *The Eleventh International Conference on Learning Representations*, 2022.

[Dua *et al.*, 2019] Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. Drop: A reading comprehension benchmark requiring discrete reasoning over paragraphs. *arXiv preprint arXiv:1903.00161*, 2019.

[Kaplan *et al.*, 2020] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[Kojima *et al.*, 2022] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213, 2022.

[Levy *et al.*, 2022] Itay Levy, Ben Bogin, and Jonathan Berant. Diverse demonstrations improve in-context compositional generalization. *arXiv preprint arXiv:2212.06800*, 2022.

[McCracken and Reilly, 2003] Daniel D McCracken and Edwin D Reilly. Backus-naur form (bnf). In *Encyclopedia of Computer Science*, pages 129–131. 2003.

[Nye *et al.*, 2021] Maxwell Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, et al. Show your work: Scratchpads for intermediate computation with language models. *arXiv preprint arXiv:2112.00114*, 2021.

[Poesia *et al.*, 2022] Gabriel Poesia, Oleksandr Polozov, Vu Le, Ashish Tiwari, Gustavo Soares, Christopher Meek, and Sumit Gulwani. Synchromesh: Reliable code generation from pre-trained language models. *arXiv preprint arXiv:2201.11227*, 2022.

[Qin *et al.*, 2023] Yujia Qin, Shengding Hu, Yankai Lin, Weize Chen, Ning Ding, Ganqu Cui, Zheni Zeng, Yufei Huang, Chaojun Xiao, Chi Han, et al.

Tool learning with foundation models. *arXiv preprint arXiv:2304.08354*, 2023.

[Qiu *et al.*, 2022] Linlu Qiu, Peter Shaw, Panupong Pasupat, Tianze Shi, Jonathan Herzig, Emily Pitler, Fei Sha, and Kristina Toutanova. Evaluating the impact of model scale for compositional generalization in semantic parsing. *arXiv preprint arXiv:2205.12253*, 2022.

[Rae *et al.*, 2021] Jack W Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, et al. Scaling language models: Methods, analysis & insights from training gopher. *arXiv preprint arXiv:2112.11446*, 2021.

[Schick *et al.*, 2024] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36, 2024.

[Scholak *et al.*, 2021] Torsten Scholak, Nathan Schucher, and Dzmitry Bahdanau. Picard: Parsing incrementally for constrained auto-regressive decoding from language models. *arXiv preprint arXiv:2109.05093*, 2021.

[Seipel *et al.*, 2018] Dietmar Seipel, Falco Nogatz, and Salvador Abreu. Domain-specific languages in prolog for declarative expert knowledge in rules and ontologies. *Computer languages, systems & structures*, 51:102–117, 2018.

[Smith *et al.*, 2022] Shaden Smith, Mostofa Patwary, Brandon Norick, Patrick LeGresley, Samyam Rajbhandari, Jared Casper, Zhun Liu, Shrimai Prabhumoye, George Zerveas, Vijay Korthikanti, et al. Using deepspeed and megatron to train megatron-turing nlg 530b, a large-scale generative language model. *arXiv preprint arXiv:2201.11990*, 2022.

[Suzgun *et al.*, 2022] Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V Le, Ed H Chi, Denny Zhou, et al. Challenging big-bench tasks and whether chain-of-thought can solve them. *arXiv preprint arXiv:2210.09261*, 2022.

[Wang *et al.*, 2022] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.

[Wang *et al.*, 2023] Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023.

[Wang *et al.*, 2024] Bailin Wang, Zi Wang, Xuezhi Wang, Yuan Cao, Rif A Saurous, and Yoon Kim. Grammar prompting for domain-specific language generation with large language models. *Advances in Neural Information Processing Systems*, 36, 2024.

[Wei *et al.*, 2022] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

[Work,] What Makes In-Context Learning Work. Rethinking the role of demonstrations: What makes in-context learning work?

[Xie *et al.*, 2023] Yuxi Xie, Kenji Kawaguchi, Yiran Zhao, Xu Zhao, Min-Yen Kan, Junxian He, and Qizhe Xie. Decomposition enhances reasoning via self-evaluation guided decoding. *arXiv preprint arXiv:2305.00633*, 2023.

[Xu *et al.*, 2022] Frank F Xu, Uri Alon, Graham Neubig, and Vincent Josua Hellendoorn. A systematic evaluation of large language models of code. In *Proceedings of the 6th ACM SIGPLAN International Symposium on Machine Programming*, pages 1–10, 2022.

[Yao *et al.*, 2022] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

[Ye *et al.*, 2020] Xi Ye, Qiaochu Chen, Isil Dillig, and Greg Durrett. Benchmarking multimodal regex synthesis with complex structures. *arXiv preprint arXiv:2005.00663*, 2020.